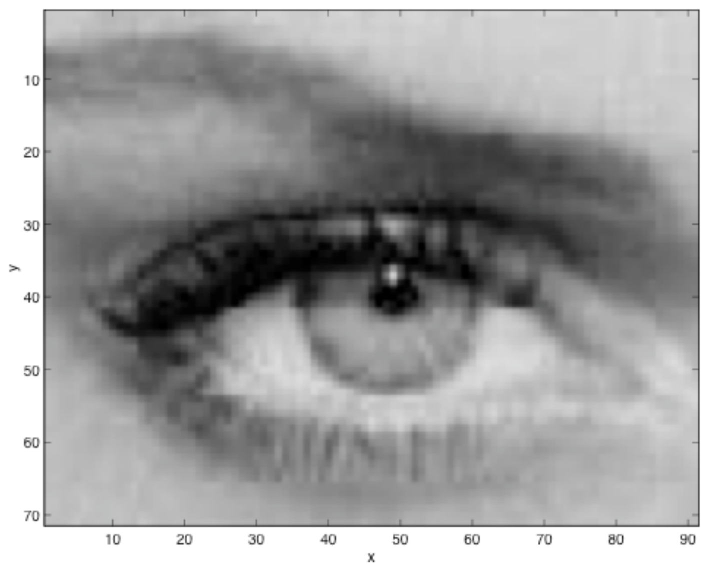
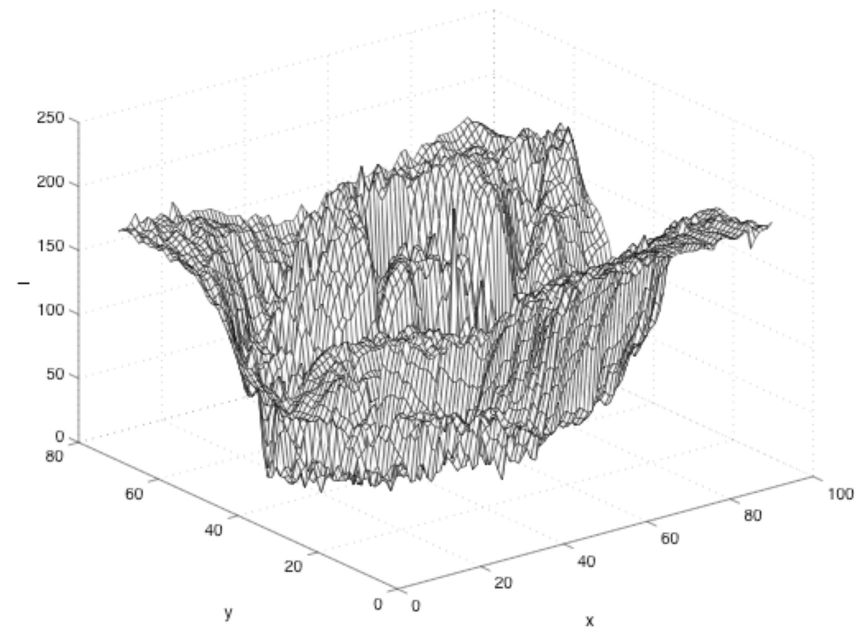
Image Primitives and Correspondence

Image



Brightness values



I(x,y)

Image Features

Local, meaningful, detectable parts of the image.

- Edge detection
- Line detection
- Corner detection

Motivation

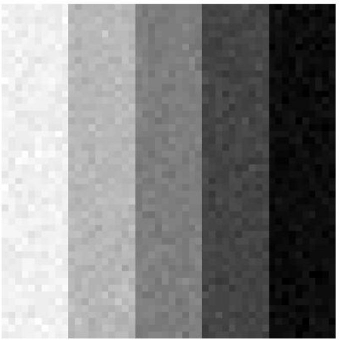
- Information content high
- Invariant to change of view point, illumination
- Reduces computational burden
- Uniqueness
- Can be tuned to a task at hand

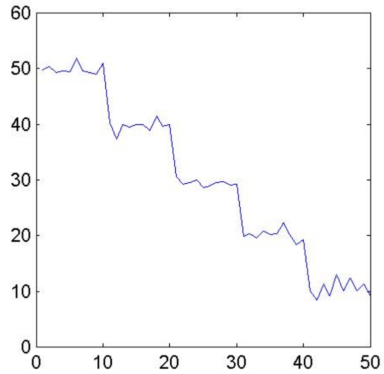
Filetring and Image Features

Given a noisy image

How do we reduce noise ?
How do we find useful features ?

Today:
- Filtering
- Point-wise operations
- Edge detection





Moving average

- Let’s replace each pixel with a *weighted* average of its neighborhood
- The weights are called the *filter kernel*
- What are the weights for the average of a 3x3 neighborhood?

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

“box filter”

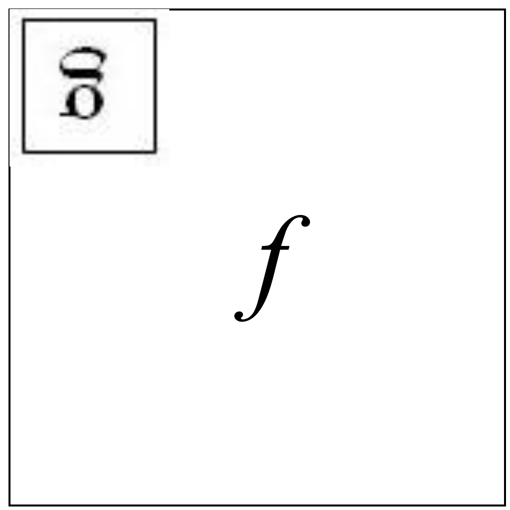
Source: D. Lowe

Defining convolution

- Let f be the image and g be the kernel. The output of convolving f with g is denoted $f * g$.

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$

Convention:
kernel is "flipped"

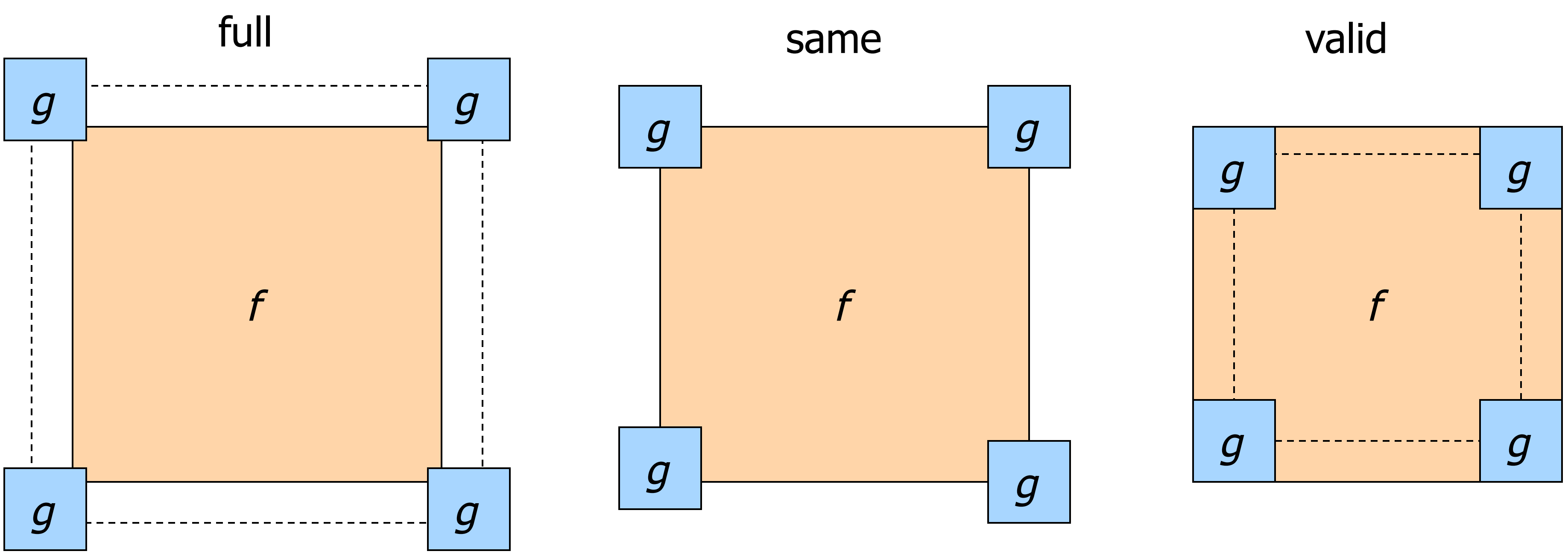


- MATLAB functions: conv2, filter2, imfilter

Source: F. Durand

Annoying details

- What is the size of the output?
- MATLAB: filter2(g, f, *shape*)
  - *shape* = 'full': output size is sum of sizes of f and g
  - *shape* = 'same': output size is same as f
  - *shape* = 'valid': output size is difference of sizes of f and g



Key properties

- **Linearity:** filter(f_1 + f_2) = filter(f_1) + filter(f_2)
- **Shift invariance:** same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

Averaging filter 1-D example

$$g[x] = \sum_{k=-\infty}^{\infty} f[k]h[x-k]$$

$$f[x] = [...0,0,2,-2,2,0,0,...] \quad h[x] = \tfrac{1}{3}[1,1,1]$$

$h[-1] = \frac{1}{3}, h[0] = \frac{1}{3}, h[1] = \frac{1}{3}$ and 0 everywhere else

$$f[-1] = -2, f[0] = 2, f[1] = -2$$

Box filter $$g[x] = \sum_{k=-1}^{1} f[k]h[x-k]$$

Ex. cont.

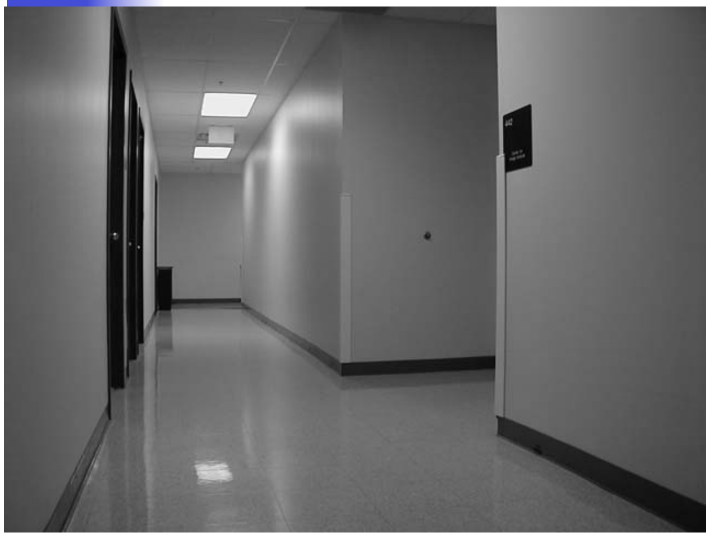
$$g[-1] = f[-1]h[-1-1] + f[0]h[-1] + f[1]h[0]$$

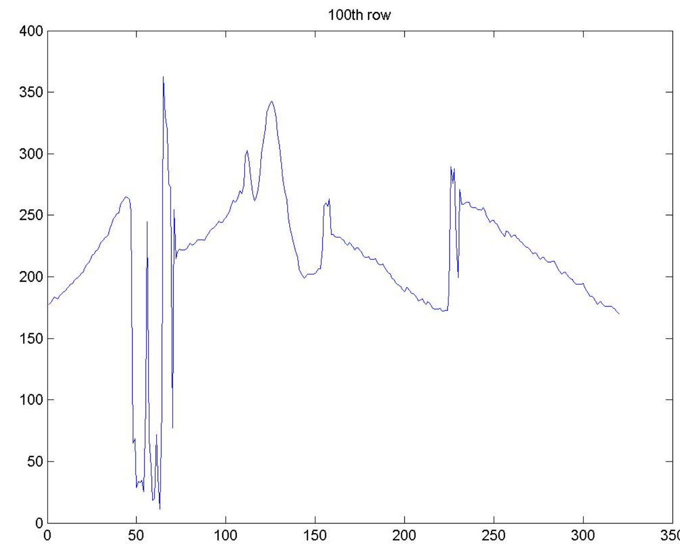
$$g[0] = f[-1]h[-1] + f[0]h[0] + f[1]h[1]$$

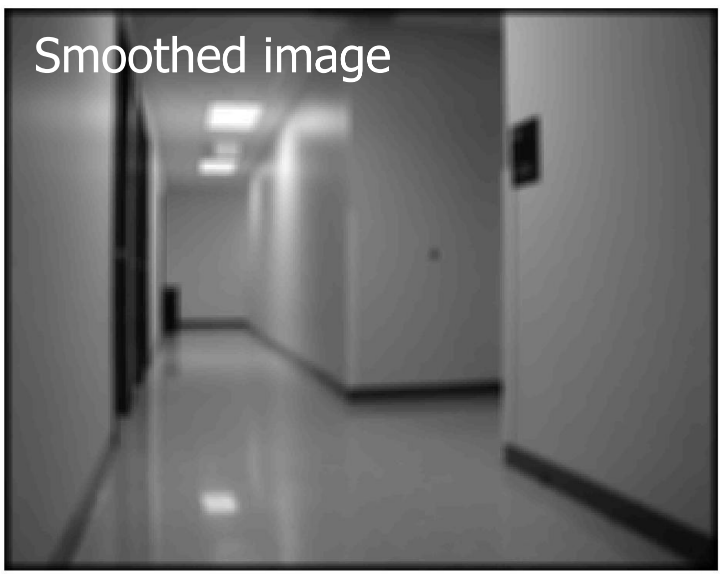
Averaging filter center pixel weighted more

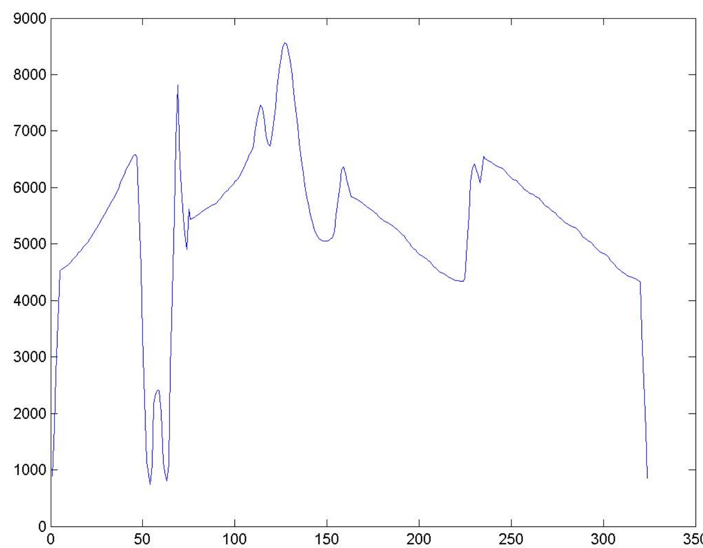
$$h[x] = [0.25, 0.5, 0.25]$$

Averaging filter









Convolution in 2D

$$g[x,y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} f[k,l]h[x-k,y-l]$$

f

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

h

1/9

1	1	1
1	1	1
1	1	1

g

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) = 1/9.(90) = 10

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 0x1 + 0x1 + 11x1 + 1x1 + 0x1 + 10x1 + 0x1 + 2x1) =

1/9.(34) = 3.7778

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	$\boxed{99}$	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X					X
X				$\boxed{20}$	X
X	X	X	X	X	X

1/9.(10x1 + 9x1 + 11x1 + 9x1 + 99x1 + 11x1 + 11x1 + 10x1 + 10x1) =
1/9.(180) = 20

Example:

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10	7	4	1	X
X					X
X			18		X
X				20	X
X	X	X	X	X	X

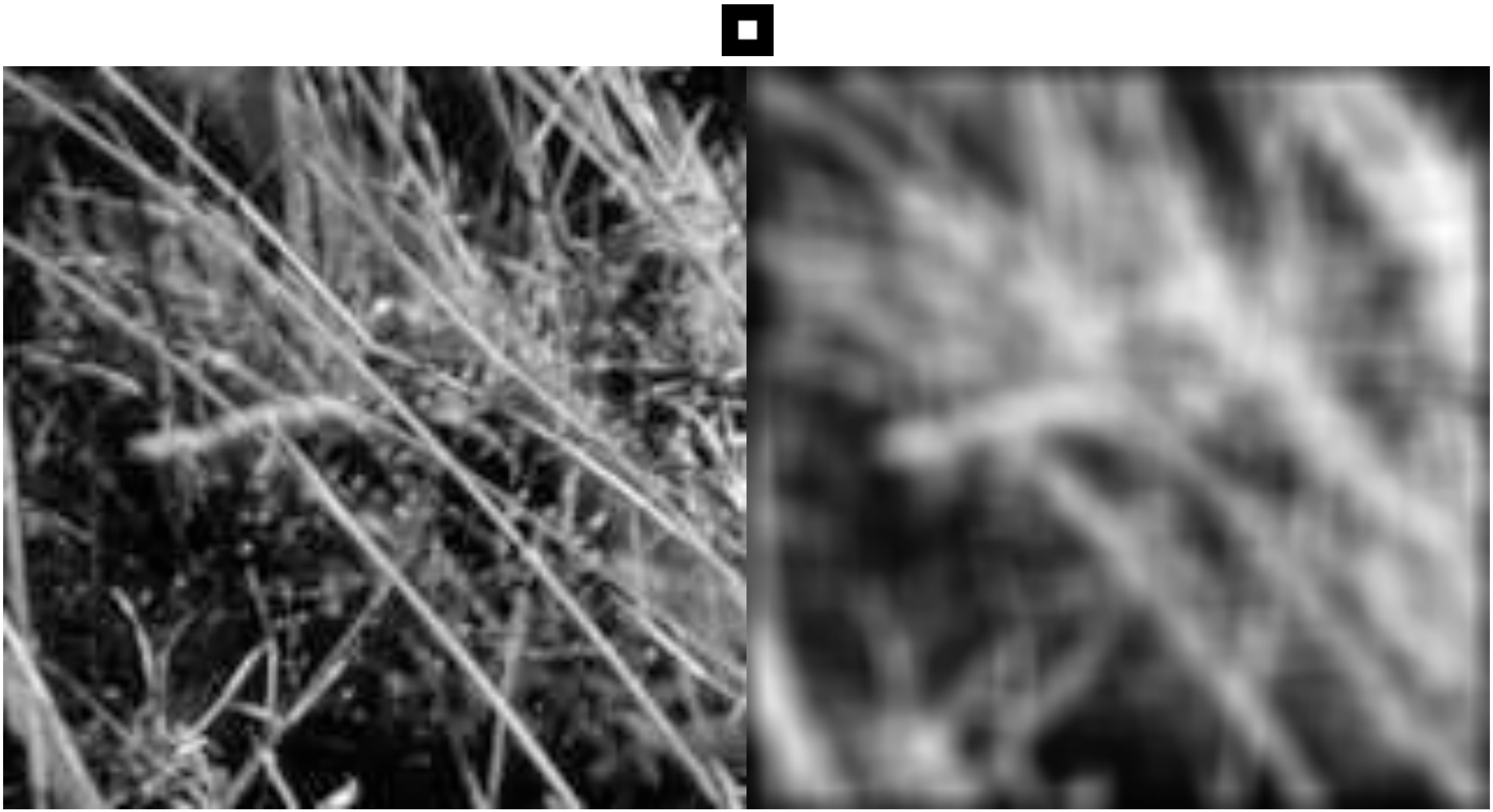
1/9.(10x1 + 0x1 + 2x1 + 9x1 + 10x1 + 9x1 + 11x1 + 9x1 + 99x1) =

1/9.(159) = 17.6667

How big should the mask be?

- The bigger the mask,
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.
  - In Matlab function **conv, conv2**

Example: Smoothing by Averaging



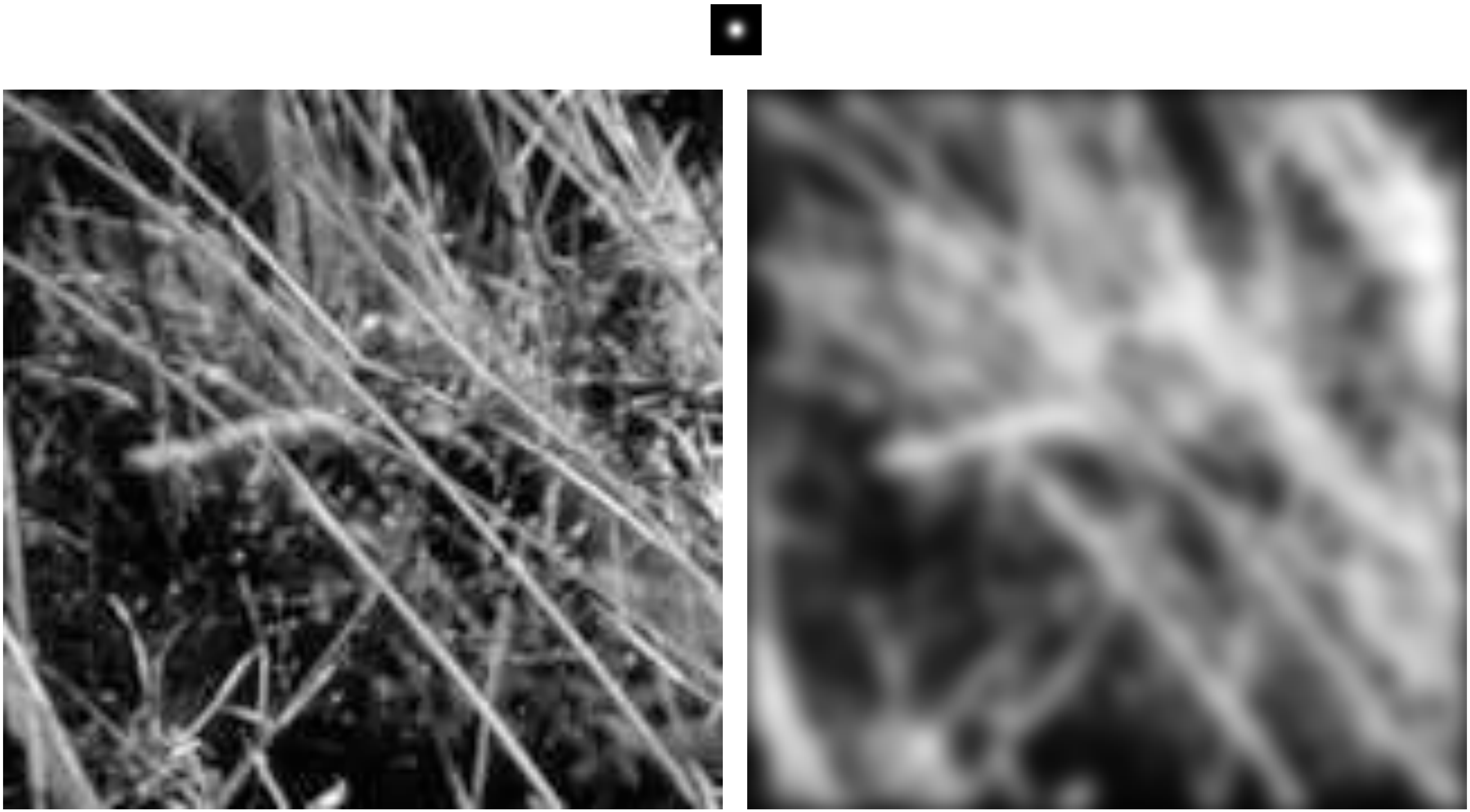
Gaussian Filter

- A particular case of averaging
  - The coefficients are samples of a 1D Gaussian.
  - Gives more weight at the central pixel and less weights to the neighbors.
  - The further away the neighbors, the smaller the weight.

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}},$$

Sample from the continuous Gaussian

Smoothing with a Gaussian



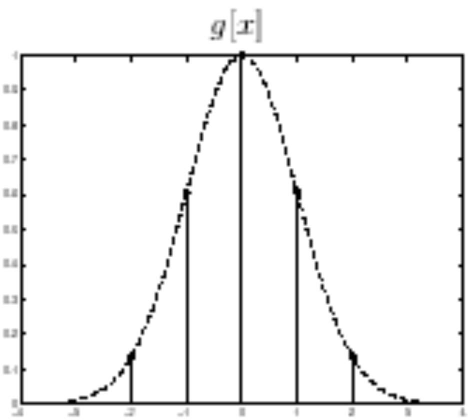
How big should the mask be?

- The std. dev of the Gaussian σ determines the amount of smoothing.
- The samples should adequately represent a Gaussian
- For a 98.76% of the area, we need

$$m = 5\sigma$$

$$5.(1/\sigma) \leq 2\pi \Rightarrow \sigma \geq 0.796,\ m \geq 5$$

5-tap filter



$$g[x] = [0.136, 0.6065, 1.00, 0.606, 0.136]$$

Gaussian filters

- Remove “high-frequency” components from the image (low-pass filter)
- Convolution with self is another Gaussian
  - So can smooth with small-σ kernel, repeat, and get same result as larger-σ kernel would have
  - Convolving two times with Gaussian kernel with std. dev. σ
    is same as convolving once with kernel with std. dev. $\sigma\sqrt{2}$
- *Separable* kernel
  - Factors into product of two 1D Gaussians

Source: K. Grauman

Separability of the Gaussian filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Source: D. Lowe

Separability example

2D convolution
(center location only)

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix}$$

The filter factors
into a product of 1D
filters:

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \times \begin{bmatrix}1&2&1\end{bmatrix}$$

Perform convolution
along rows:

$$\begin{bmatrix}1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix} = \begin{bmatrix}&11&\\&18&\\&18&\end{bmatrix}$$

Followed by convolution
along the remaining column:

Source: K. Grauman

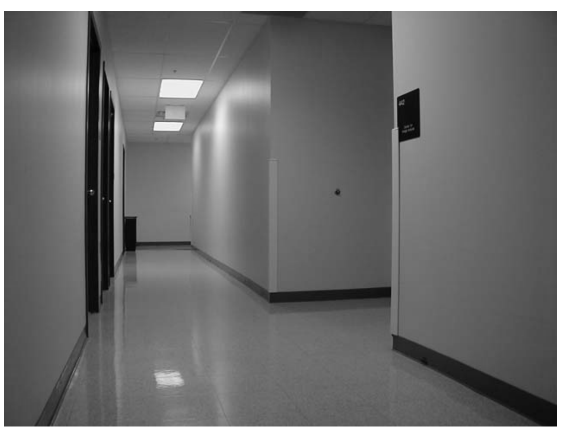
Image Smoothing

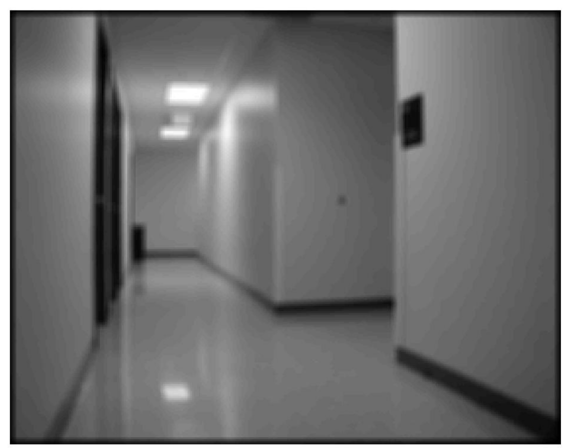
- Convolution with a 2D Gaussian filter

$$\tilde{I}(x,y) = I(x,y) * g(x,y) = I(x,y) * g(x) * g(y)$$

- Gaussian filter is separable, convolution can be accomplished as two 1-D convolutions

$$\tilde{I}[x,y] = I[x,y] * g[x,y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l]g[x-k]g[y-l]$$



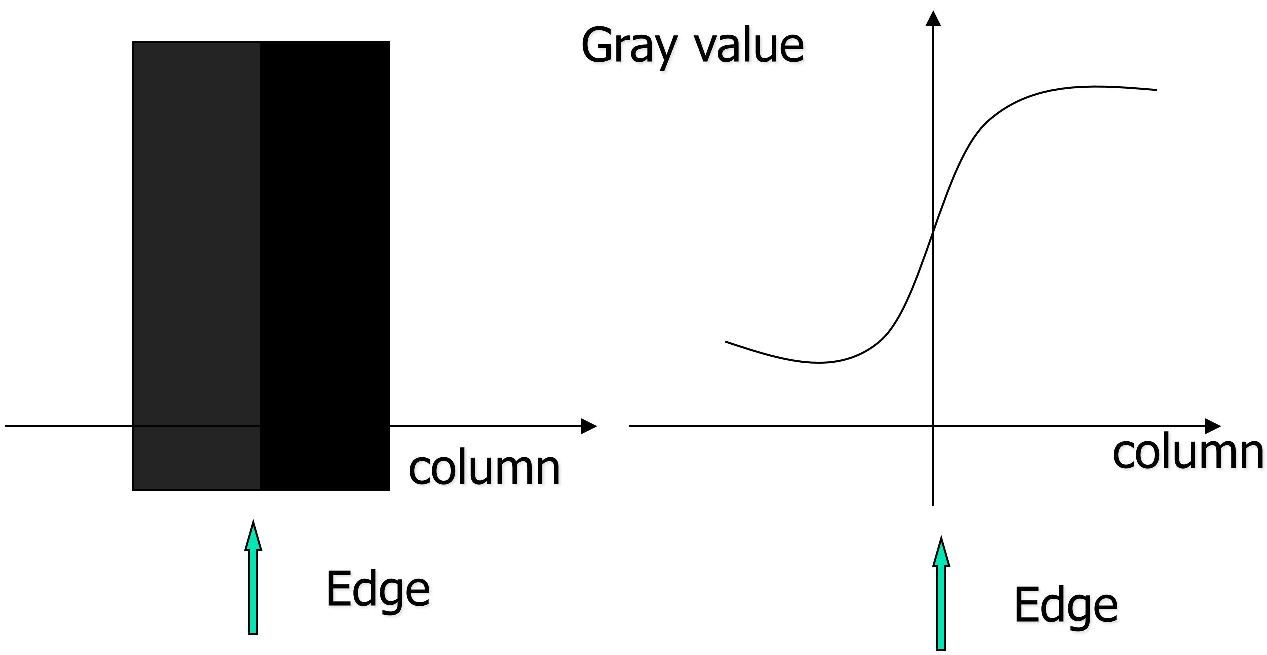


How big should the mask be?

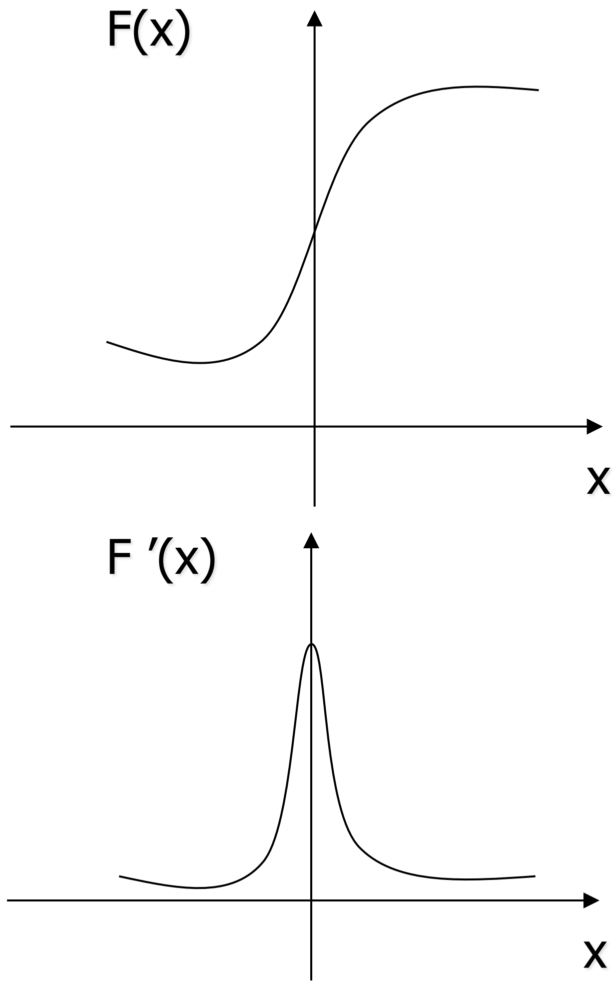
- The bigger the mask,
  - more neighbors contribute.
  - smaller noise variance of the output.
  - bigger noise spread.
  - more blurring.
  - more expensive to compute.

Edges

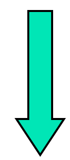
- They happen at places where the image values exhibit sharp variation



Edge detection (1D)



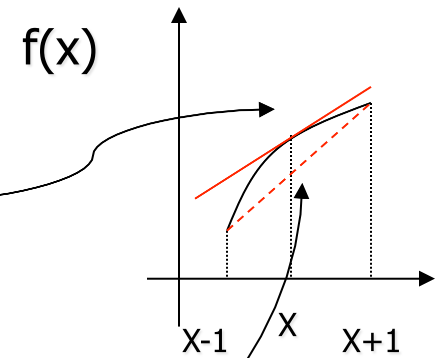
Edge= sharp variation



Large first derivative

Digital Approximation of 1st derivatives

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x+\Delta x) - f(x)}{\Delta x}$$



$$\frac{df(x)}{dx} \cong \frac{f(x+1) - f(x-1)}{2}$$

Convolve with:

-1	0	1

Edge Detection (2D)

Vertical Edges:

Convolve with:

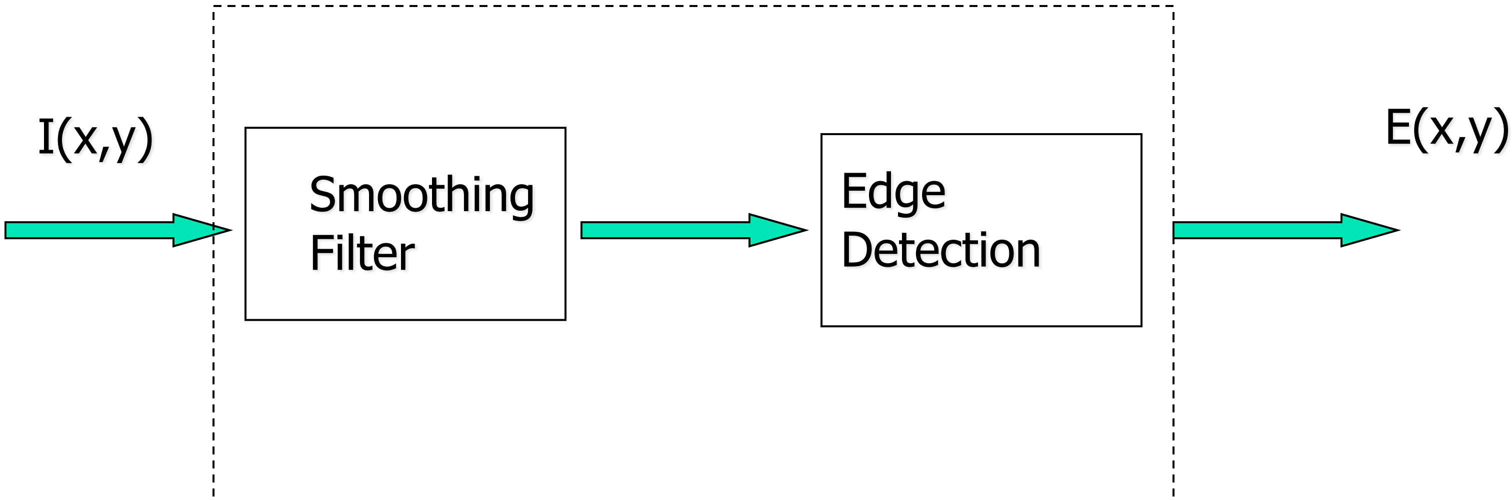
-1	0	1

Horizontal Edges:

Convolve with:

-1
0
1

Noise cleaning and Edge Detection



We need to also deal with noise
Combine Linear Filters

Noise Smoothing & Edge Detection

Convolve with:

-1	0	1
-1	0	1
-1	0	1

Noise Smoothing

Vertical Edge Detection

This mask is called the (vertical) Prewitt Edge Detector

Outer product of box filter $[1\ 1\ 1]^T$ and $[-1\ 0\ 1]$

Noise Smoothing & Edge Detection

Convolve with:

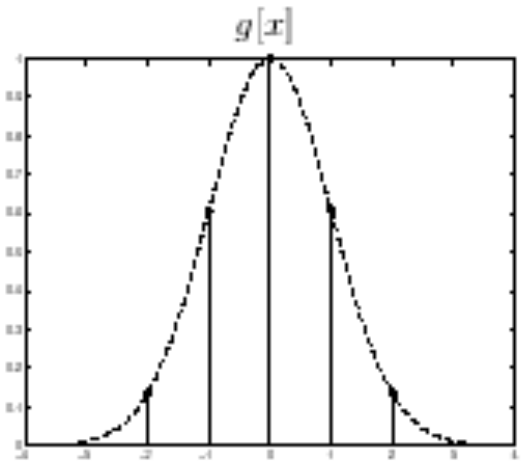
-1	-1	-1
0	0	0
1	1	1

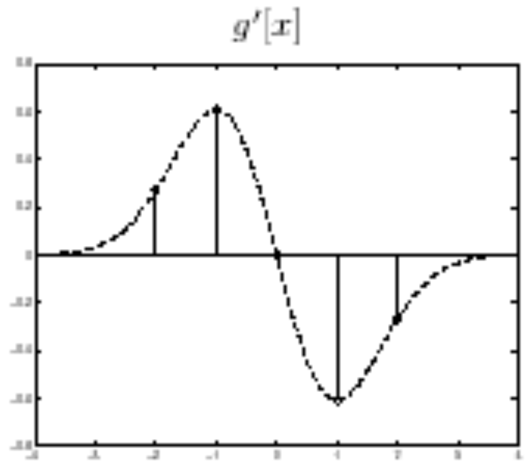
Noise Smoothing

Horizontal Edge Detection

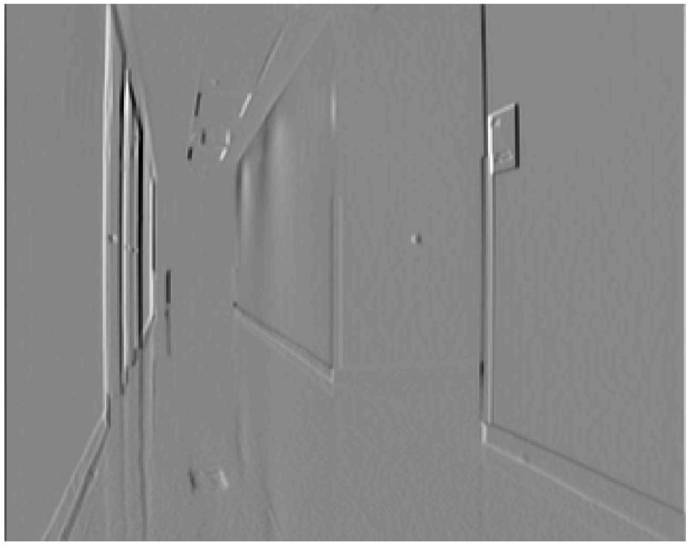
This mask is called the (horizontal) Prewitt Edge Detector

Gaussian and its derivative

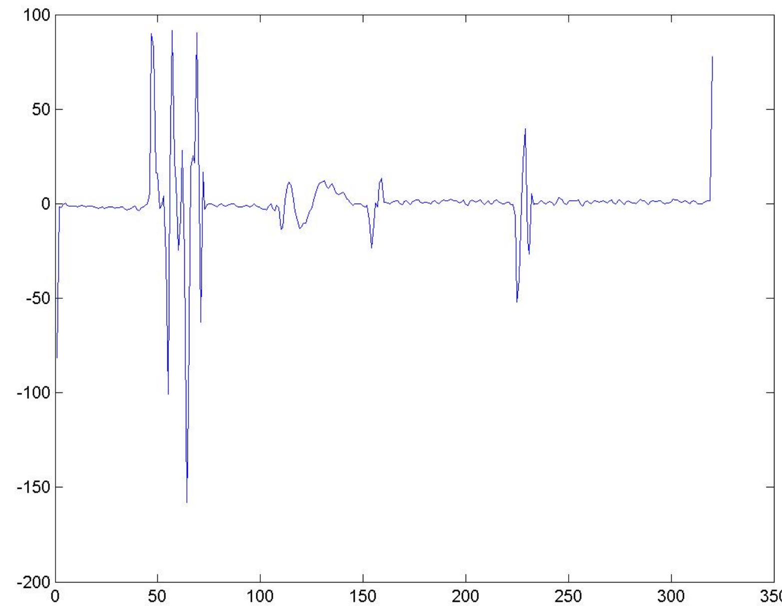




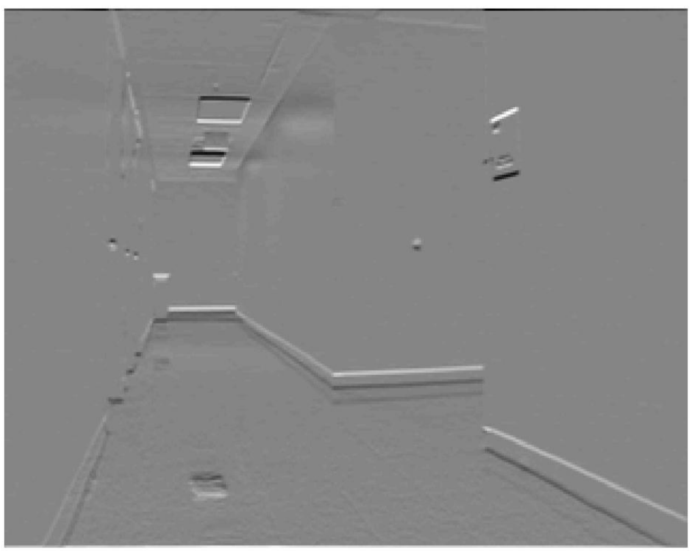
$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}, \quad g'(x) = -\frac{x}{\sigma^2\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}.$$



Vertical edges $I_x(x,y) = \frac{\partial I}{\partial x}$



First derivative - one column



$I_y(x,y) = \frac{\partial I}{\partial y}$

Horizontal edges

Gradient orientation

- Image Gradient

$$\nabla I = [\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}]$$

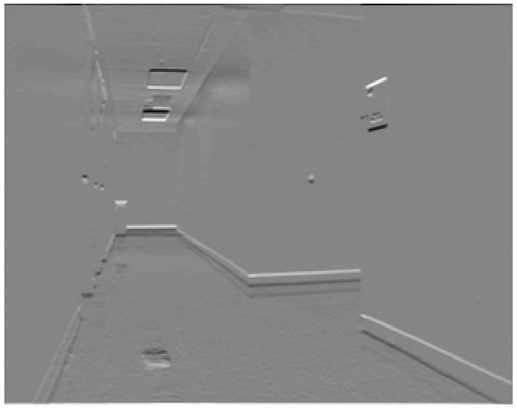
- Gradient Magnitude

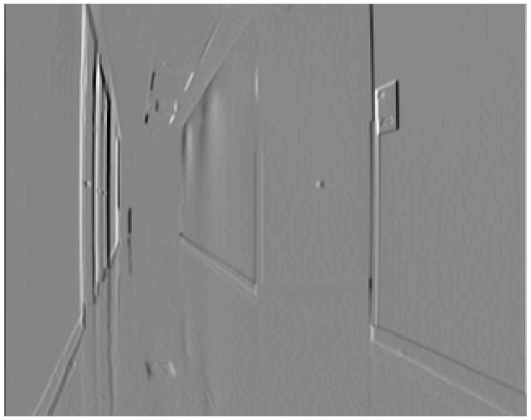
$$m = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$$

- Gradient Orientation

$$\theta = \tan^{-1}(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y})$$

Canny Edge Detector

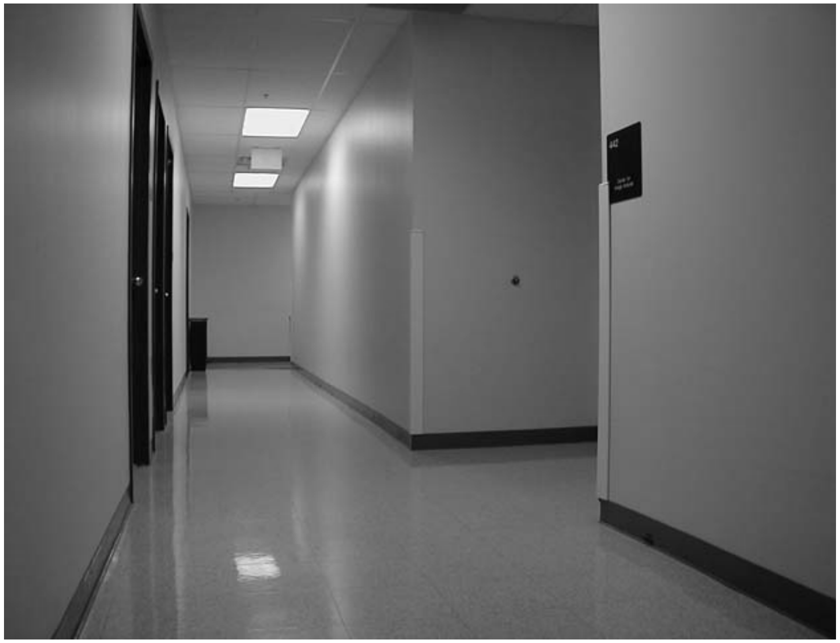




- Edge detection involves 3 steps:
  - Noise smoothing
  - Edge enhancement
  - Edge localization
- J. Canny formalized these steps to design an *optimal* edge detector
- How to go from derivatives to edges ?

Horizontal edges

Edge Detection



original image

gradient magnitude

Canny edge detector

- Compute image derivatives
- if gradient magnitude > τ and the value is a local maximum along gradient direction – pixel is an edge candidate

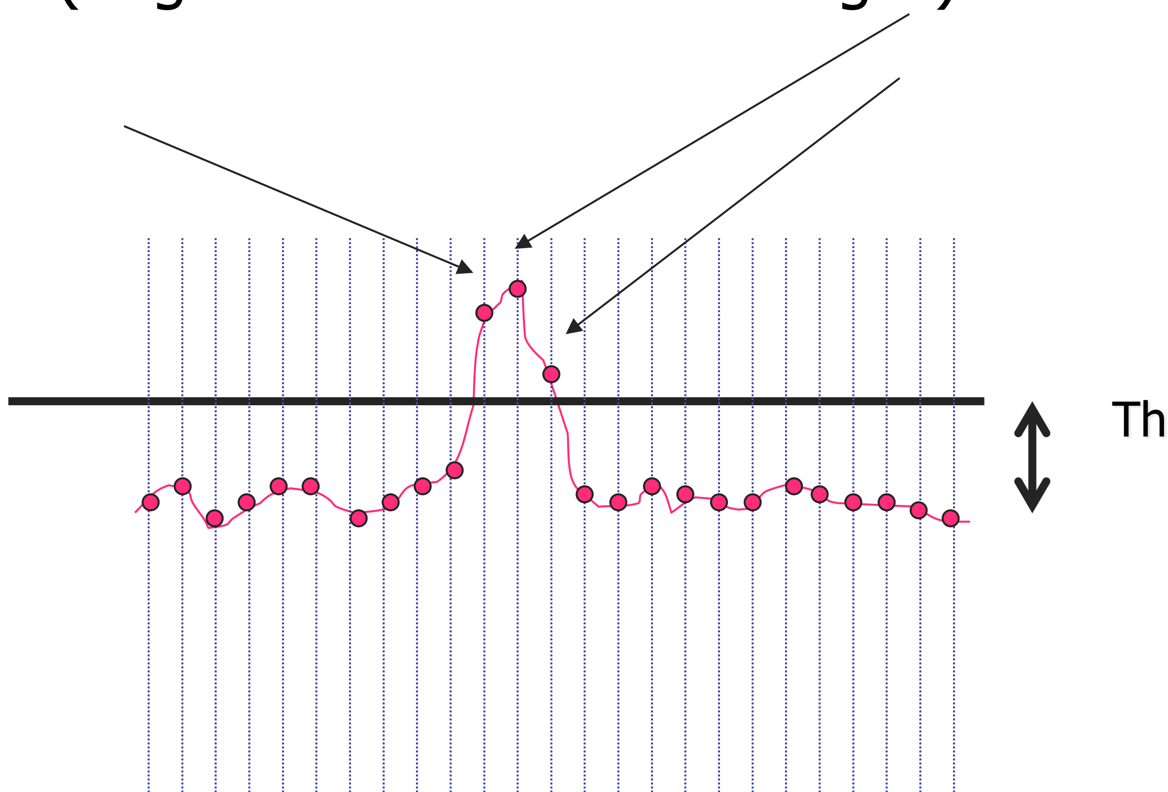
Algorithm Canny Edge detector

- The input is image I; G is a zero mean Gaussian filter (std = σ)

  1. $J = I * G$ (smoothing)
  2. For each pixel (i,j): (edge enhancement)
     - Compute the image gradient
       - $\nabla J(i,j) = (J_x(i,j), J_y(i,j))'$
     - Estimate edge strength
       - $e_s(i,j) = (J_x^2(i,j) + J_y^2(i,j))^{1/2}$
     - Estimate edge orientation
       - $e_o(i,j) = \arctan(J_x(i,j)/J_y(i,j))$
- The output are images E_s - Edge Strength - Magnitude
- and Edge Orientation E_o .

- E_s has large values at edges: Find local maxima

- … but it also may have wide ridges around the local maxima (large values *around* the edges)

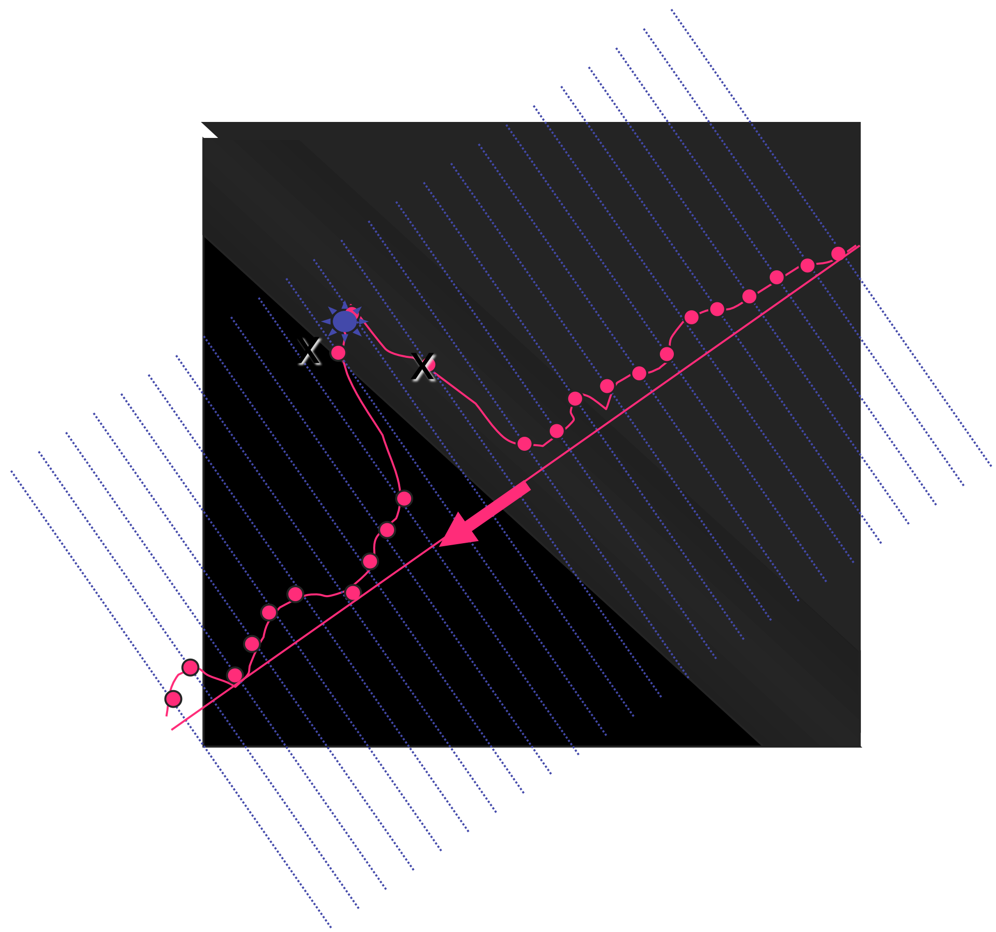


NONMAX_SUPRESSION

Edge orientation

- The inputs are E_s & E_o (outputs of CANNY_ENHANCER)
- Consider 4 directions $D=\{ 0,45,90,135\}$ wrt x
- For each pixel (i,j) do:
  1. Find the direction $d \in D$ s.t. $d \cong E_o(i,j)$ (normal to the edge)
  2. If {$E_s(i,j)$ is smaller than at least one of its neigh. along d}
     - $I_N(i,j)=0$
     - Otherwise, $I_N(i,j)= E_s(i,j)$
- The output is the thinned edge image I_N

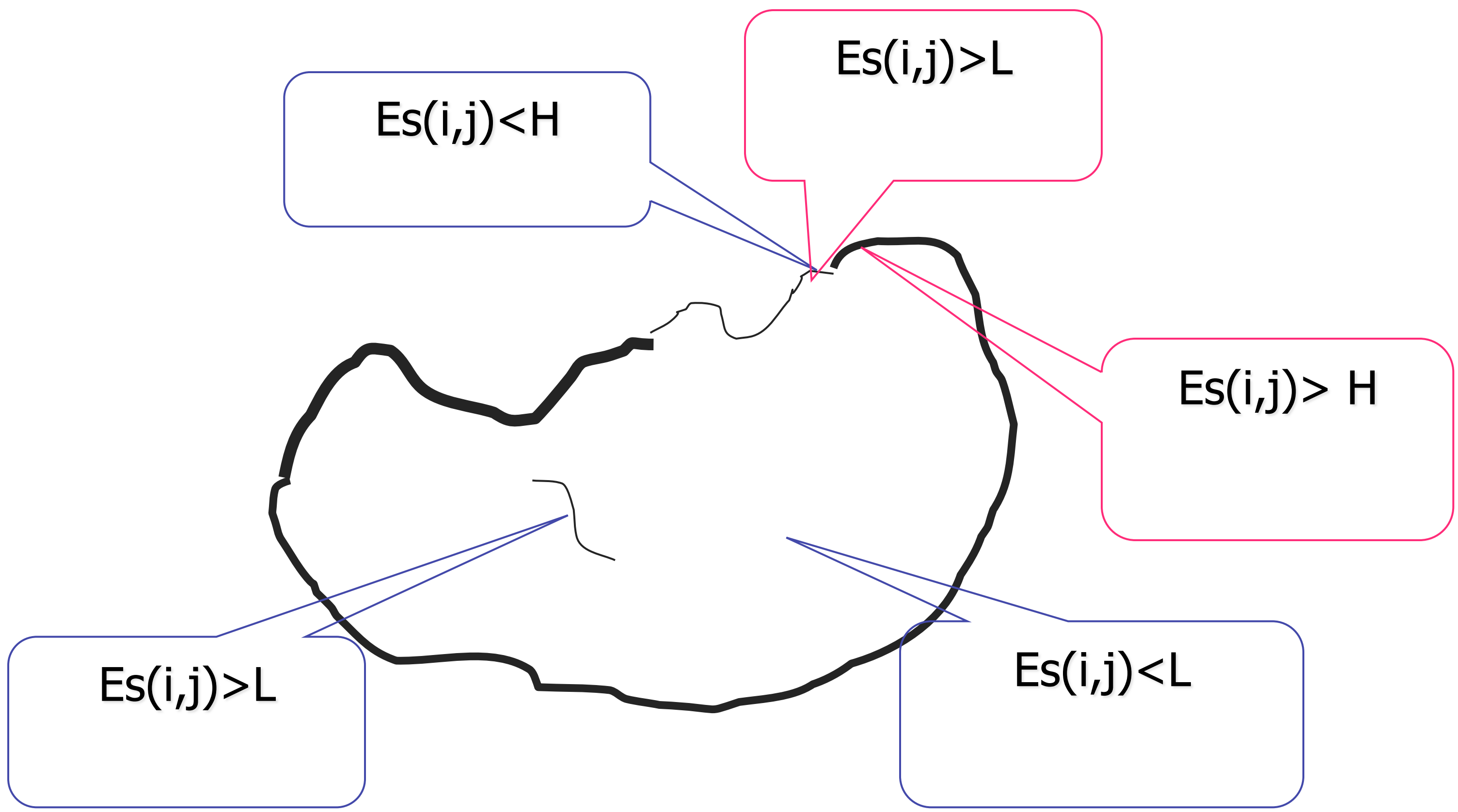
Graphical Interpretation



Thresholding

- Edges are found by thresholding the output of NONMAX_SUPRESSION
- If the threshold is too high:
  - Very few (none) edges
    - High MISDETECTIONS, many gaps
- If the threshold is too low:
  - Too many (all pixels) edges
    - High FALSE POSITIVES, many extra edges

SOLUTION: Hysteresis Thresholding



Canny Edge Detection (Example)

Original image

gap is gone

Strong + connected weak edges

Strong edges only

Weak edges

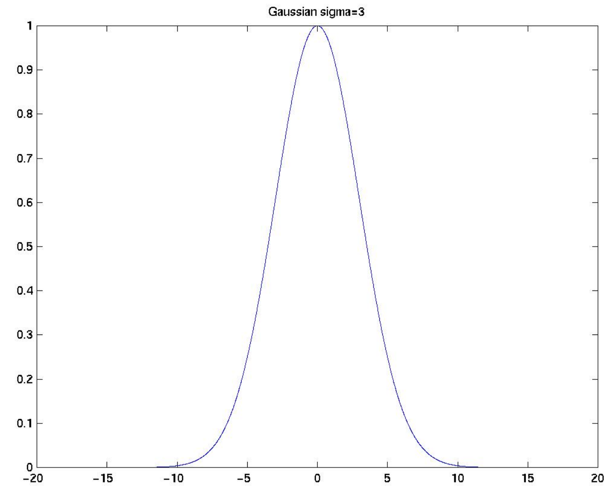
courtesy of G. Loy

LOG Filter

- First smooth (Gaussian filter),
- Then, find zero-crossings (Laplacian filter):
  - $O(x,y) = \nabla^2(I(x,y) * G(x,y))$
- Using linearity:
  - $O(x,y) = \nabla^2 G(x,y) * I(x,y)$
  - This filter is called: "Laplacian of the Gaussian" (LOG)

1D Gaussian

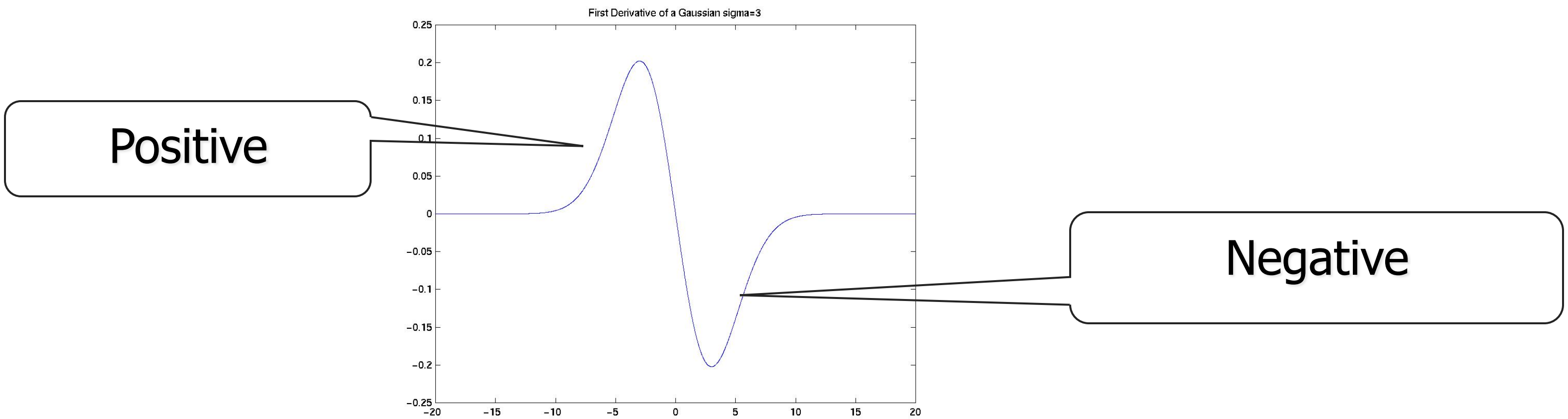
$$g(x) = e^{-\frac{x^2}{2\sigma^2}}$$



$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

First Derivative of a Gaussian

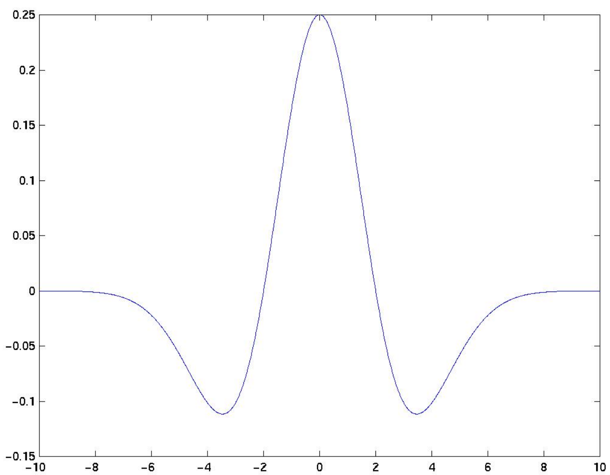
$$g'(x) = -\frac{1}{2\sigma^2} 2xe^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$



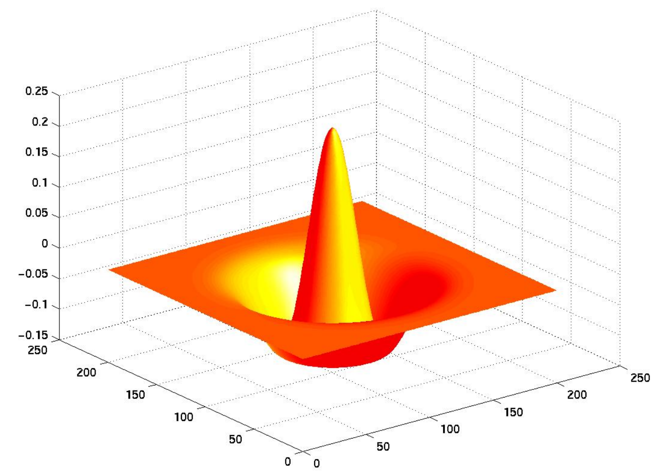
As a mask, it is also computing a difference (derivative)

Second Derivative of a Gaussian

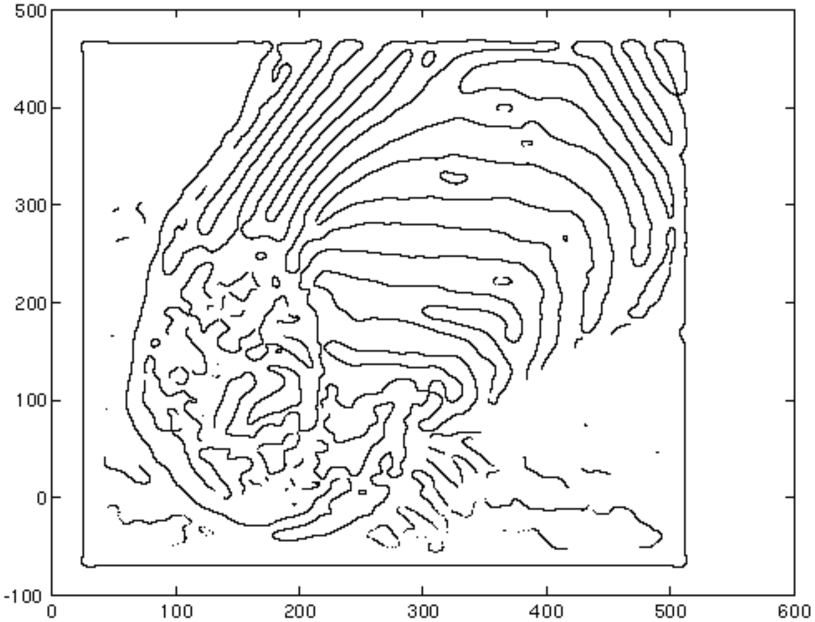
$$g''(x) = \left(\frac{x^2}{\sigma^3} - \frac{1}{\sigma}\right)e^{-\frac{x^2}{2\sigma^2}}$$



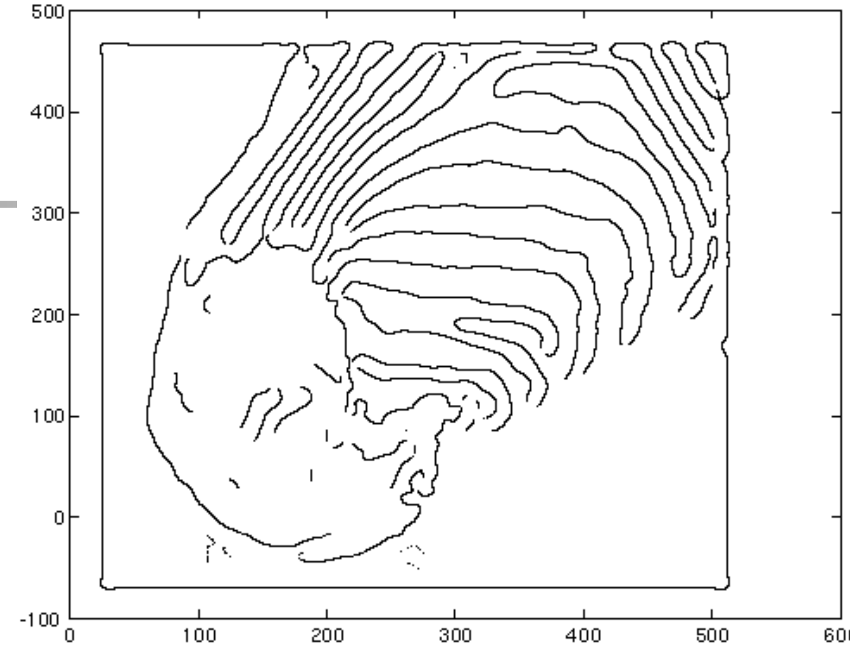
2D



"Mexican Hat"



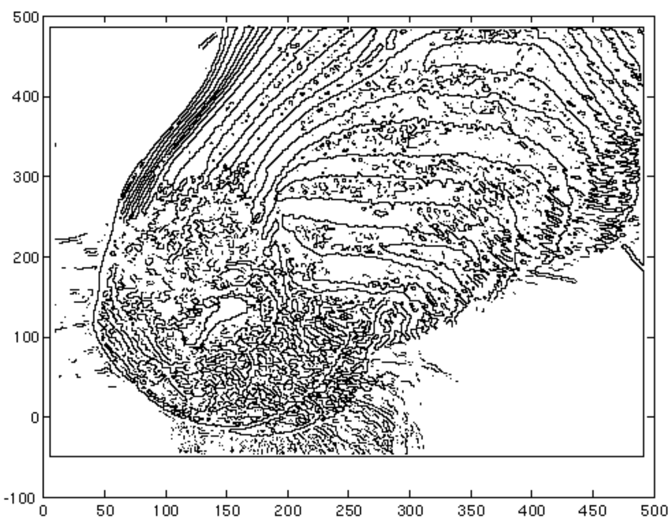
sigma=4



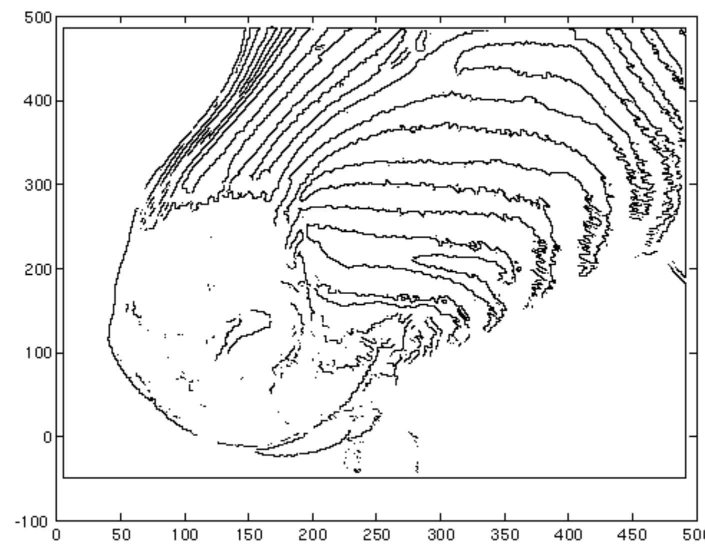
LOG zero crossings

contrast=1

contrast=4



sigma=2



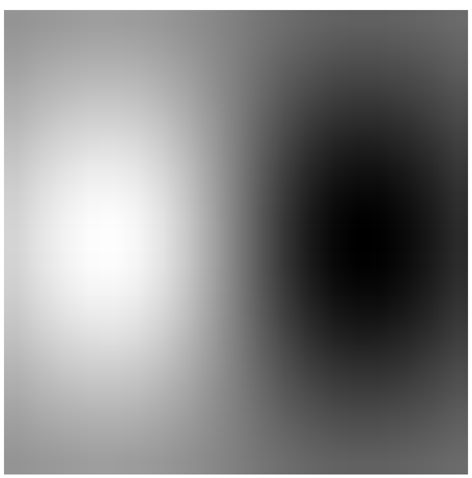
An edge is not a line...

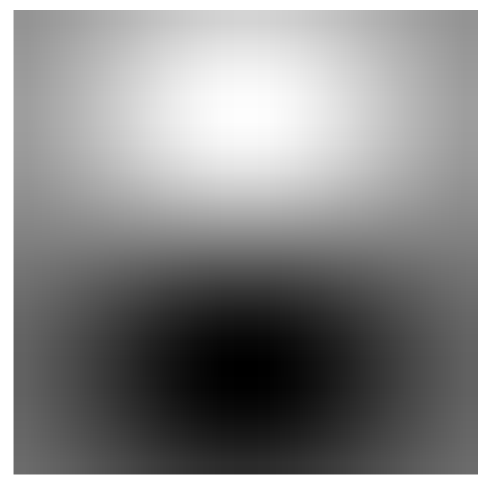
- How can we detect ***lines*** ?

Filters are templates

- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products

- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like





Robinson Compass Masks

-1	0	1
-2	0	2
-1	0	1

→

0	1	2
-1	0	1
-2	-1	0

↗

1	2	1
0	0	0
-1	-2	-1

↑

2	1	0
1	0	-1
0	-1	-2

↖

1	0	-1
2	0	-2
1	1	-1

←

0	-1	-2
-1	0	-1
2	1	0

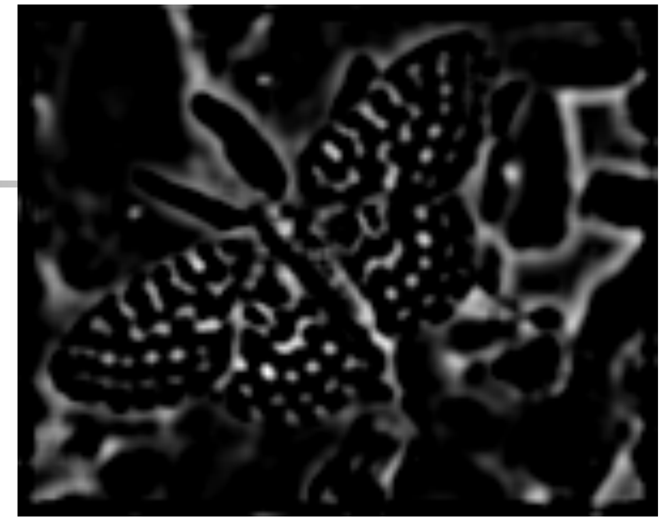
↙

-1	-2	-1
0	0	0
1	2	1

↓

-2	-1	0
-1	0	1
0	1	2

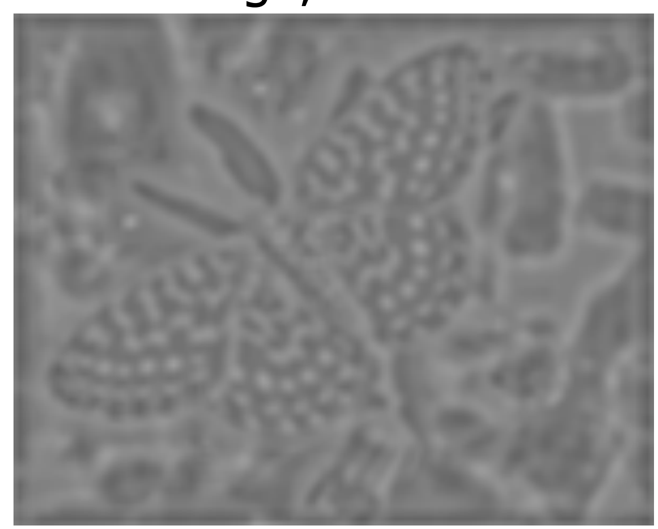
↘



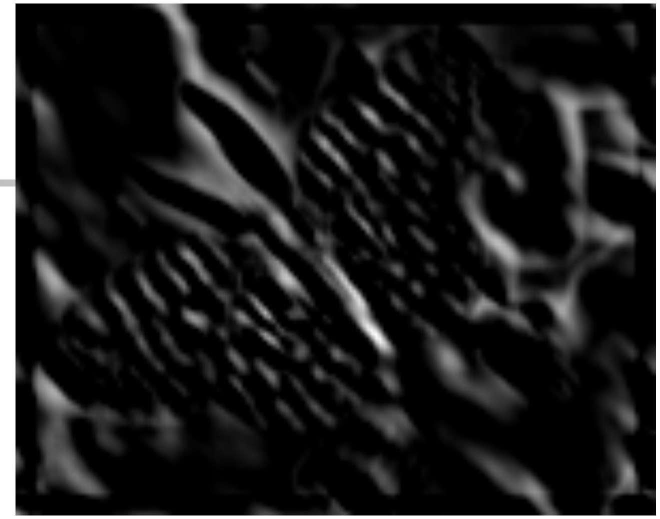
Positive responses

Zero mean image, -1:1 scale

Zero mean image, -max:max scale

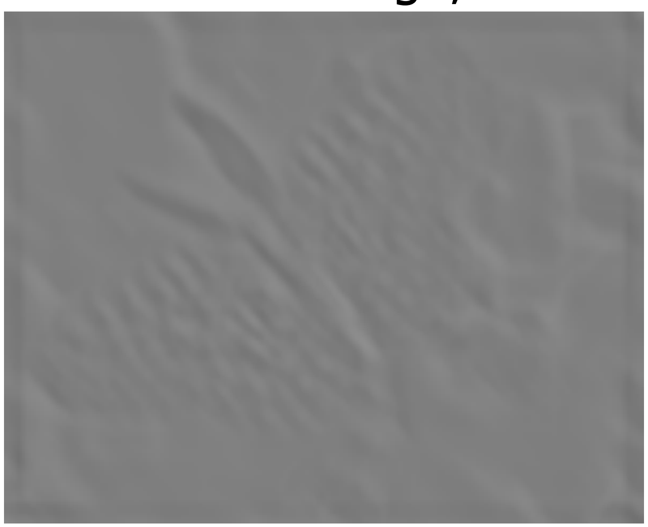


The filter is the small block at the top left corner

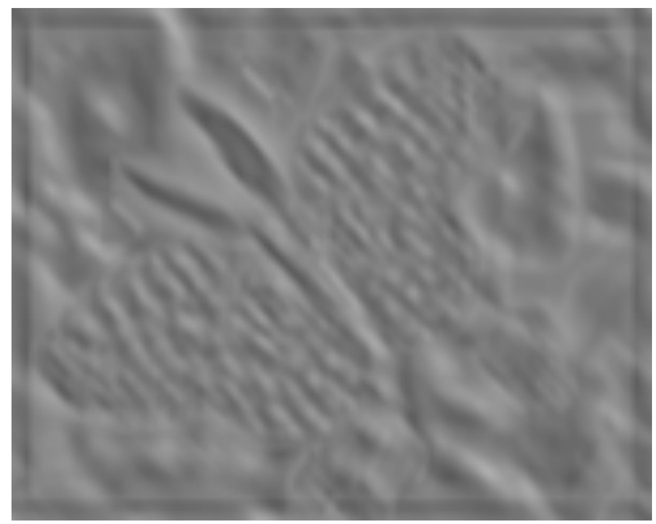


Positive responses

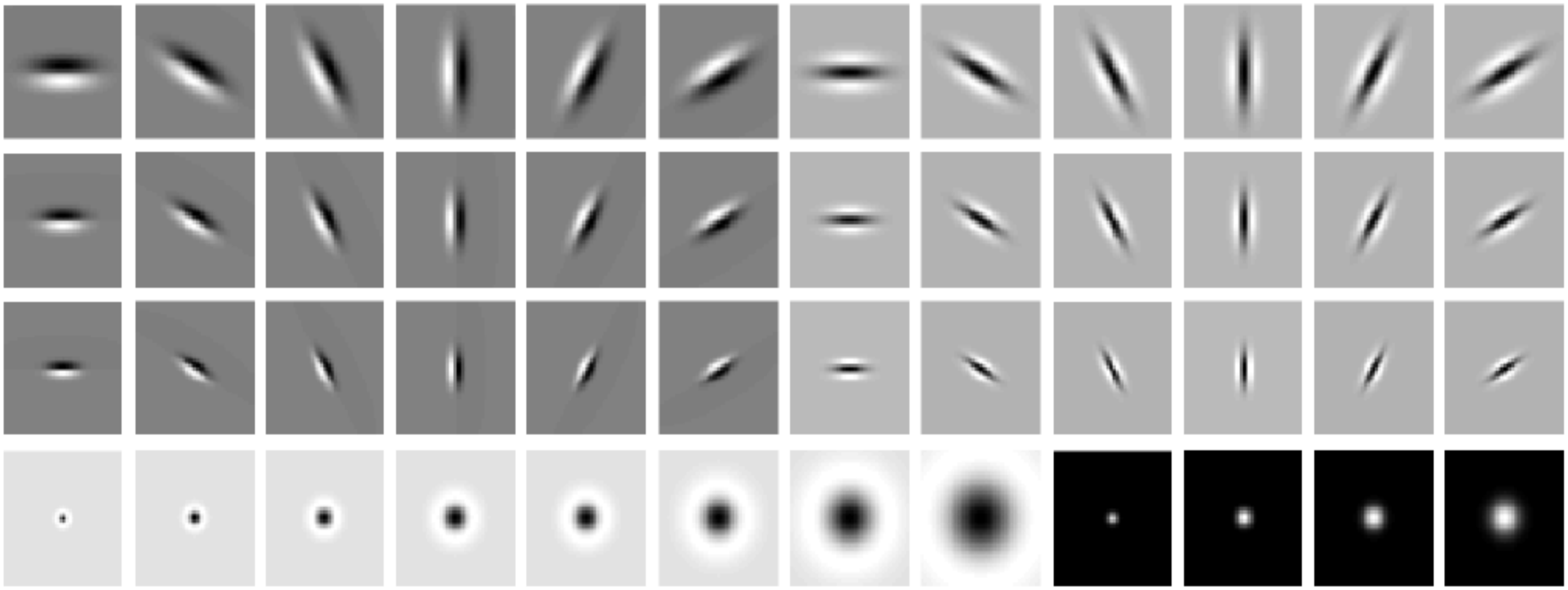
Zero mean image, -1:1 scale



Zero mean image, -max:max scale



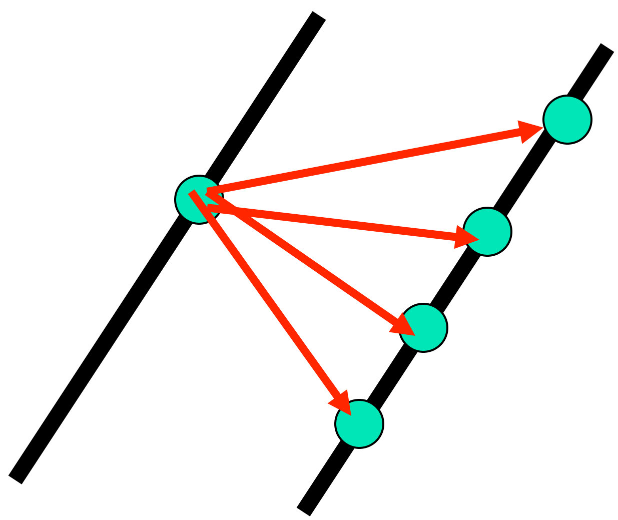
Filter Bank



Leung & Malik, Representing and Recognizing the Visual Apperance using 3D Textons, IJCV 2001

Corner Detection

- A point on a line is hard to match.



Intuition:

- Right at corner, gradient is ill defined.
- Near corner, gradient has two different values.

Formula for Finding Corners

We look at matrix:

Sum over a small region, the hypothetical corner

Gradient with respect to x, times gradient with respect to y

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

Matrix is symmetric

First, consider case where:

$$C = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

This means all gradients in neighborhood are:

(k,0) or (0, c) or (0, 0) (or off-diagonals cancel).

What is region like if:

$\lambda_1 = 0, \lambda_2 >> 0$

$\lambda_1 = 0, \lambda_2 = 0$

λ_1, λ_2 Are both large

λ_1, λ_2 Are both small

General Case:

From Linear Algebra, it follows that because C is symmetric:

$$C = R^{-1}\begin{bmatrix}\lambda_1 & 0 \\ 0 & \lambda_2\end{bmatrix}R$$

With R a rotation matrix.

So every case is like one on last slide.

Corner detection

- Filter image.
- Compute magnitude of the gradient everywhere.
- We construct C in a window.
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$.
- If they are both big, we have a corner.

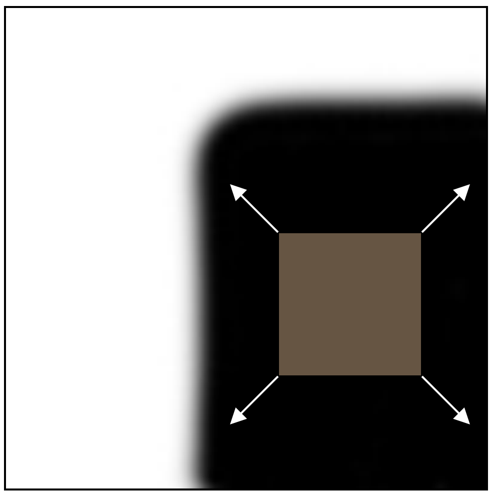
Corner detection

- Filter image.
- Compute magnitude of the gradient everywhere.
- We construct C in a window.
- Use Linear Algebra to find $\lambda 1$ and $\lambda 2$.
- If they are both big, we have a corner.

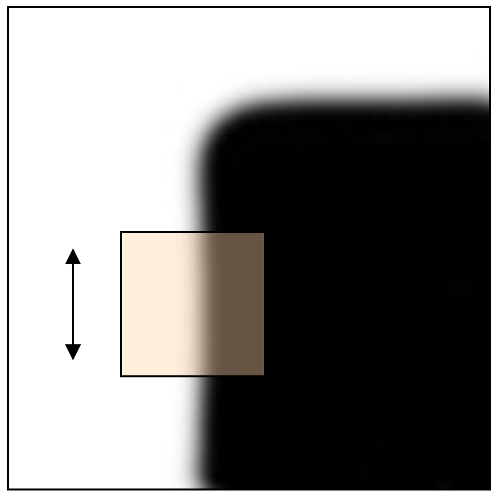
- Key property: in the region around a corner, image gradient has two or more dominant directions
- Corners are repeatable and distinctive

Corner Detection: Basic Idea

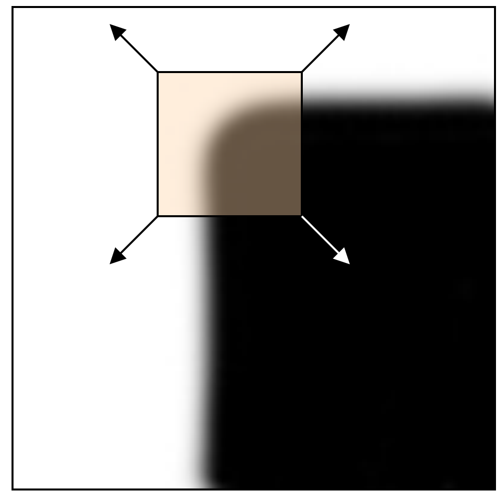
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



"flat" region:
no change in all directions



"edge":
no change along the edge direction

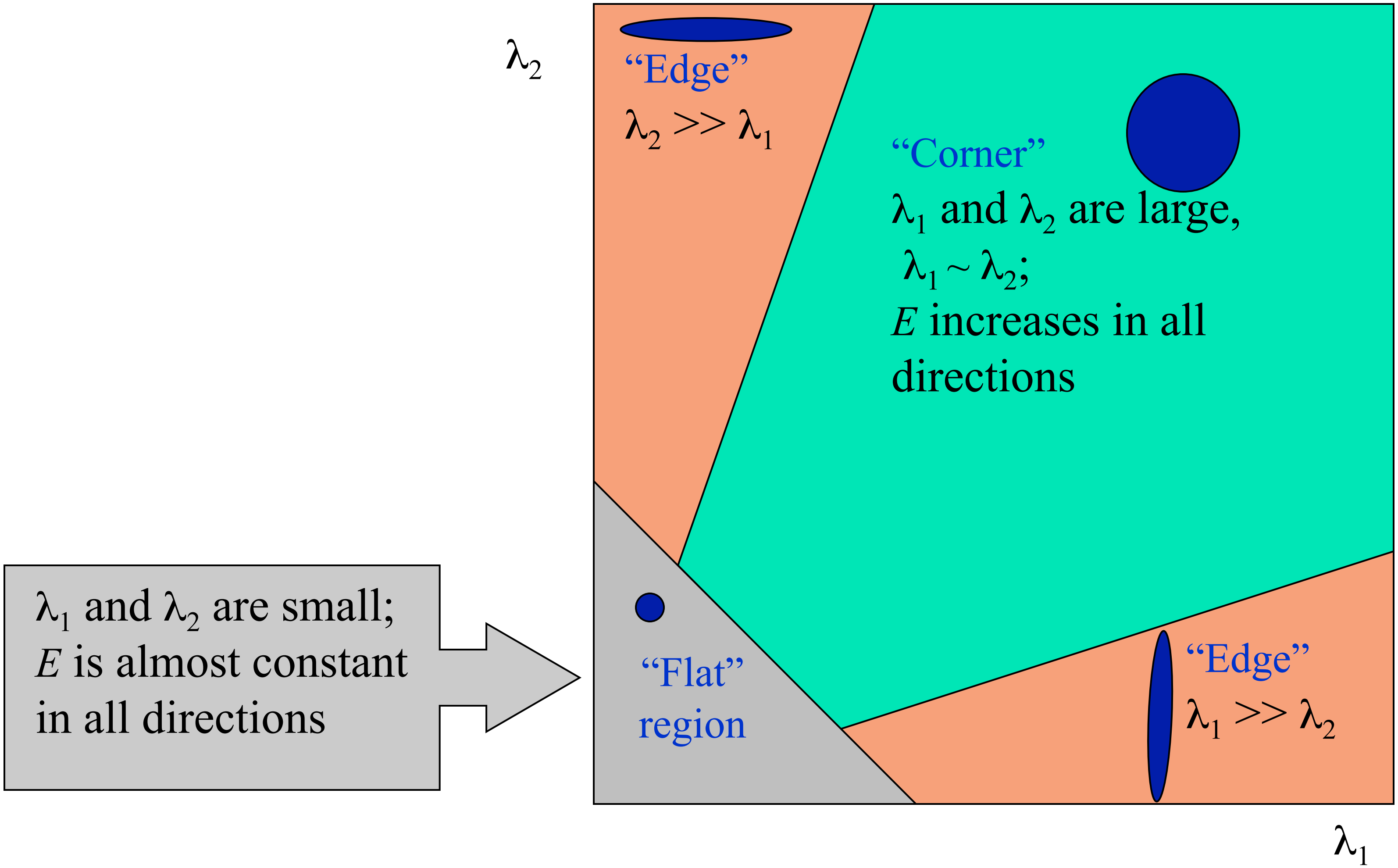


"corner":
significant change in all directions

Source: A. Efros

Interpreting the eigenvalues

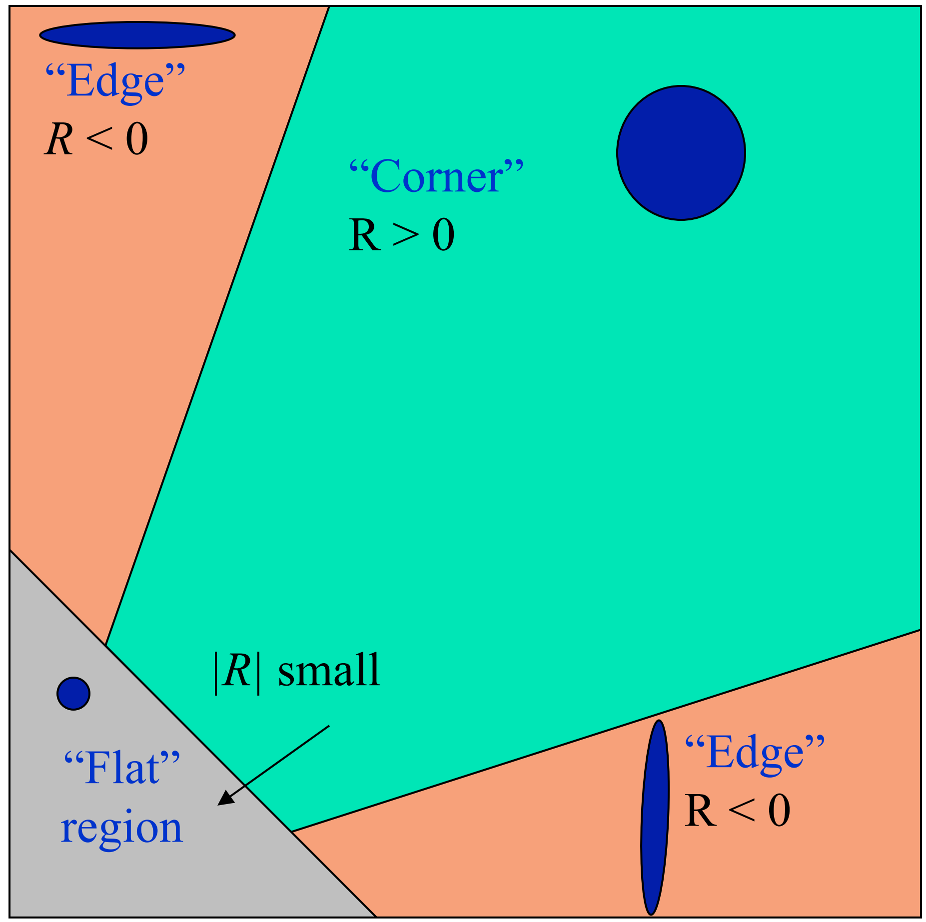
Classification of image points using eigenvalues of *C*:

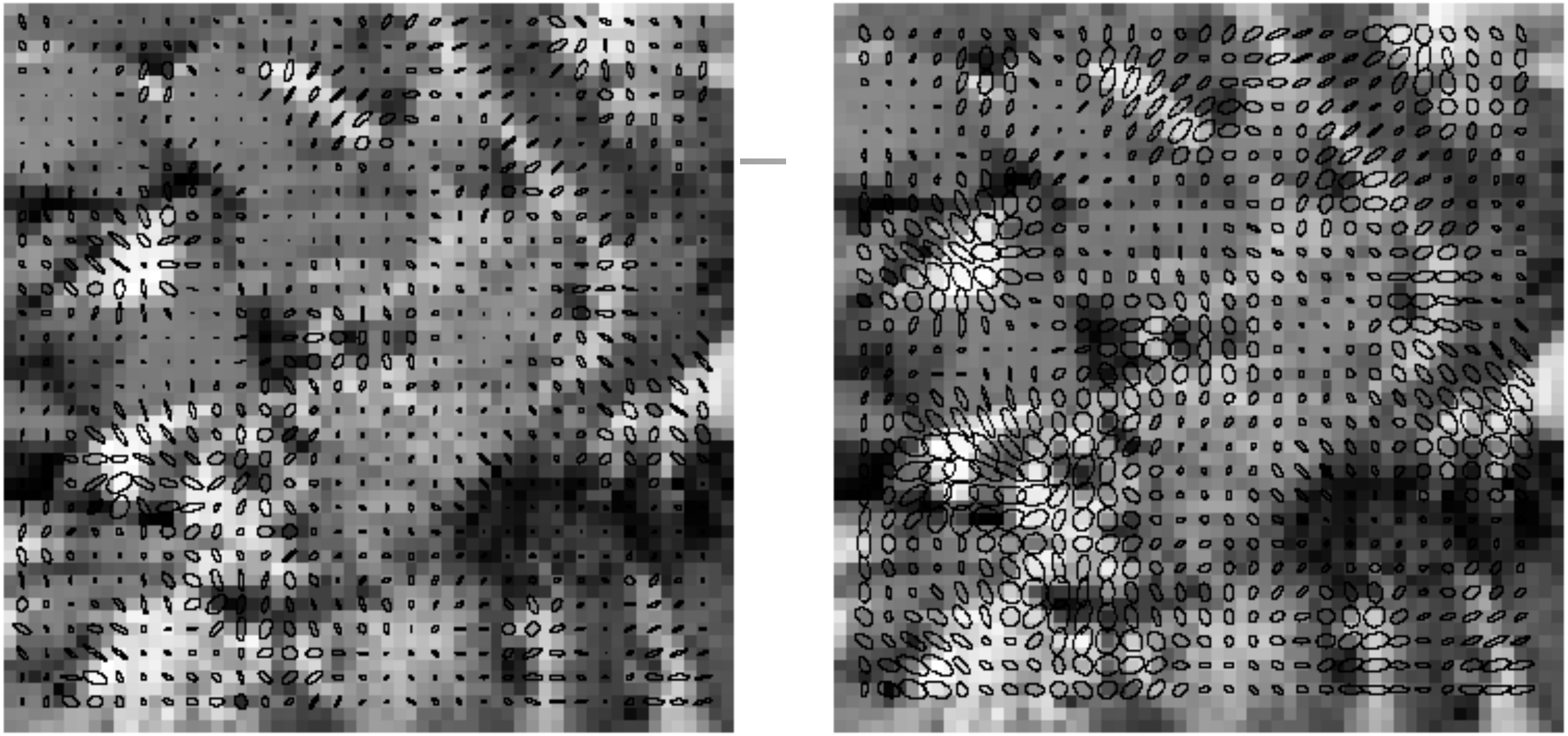


Corner response function

$$R = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1\lambda_2 - \alpha(\lambda_1 + \lambda_2)^2$$

α: constant (0.04 to 0.06)

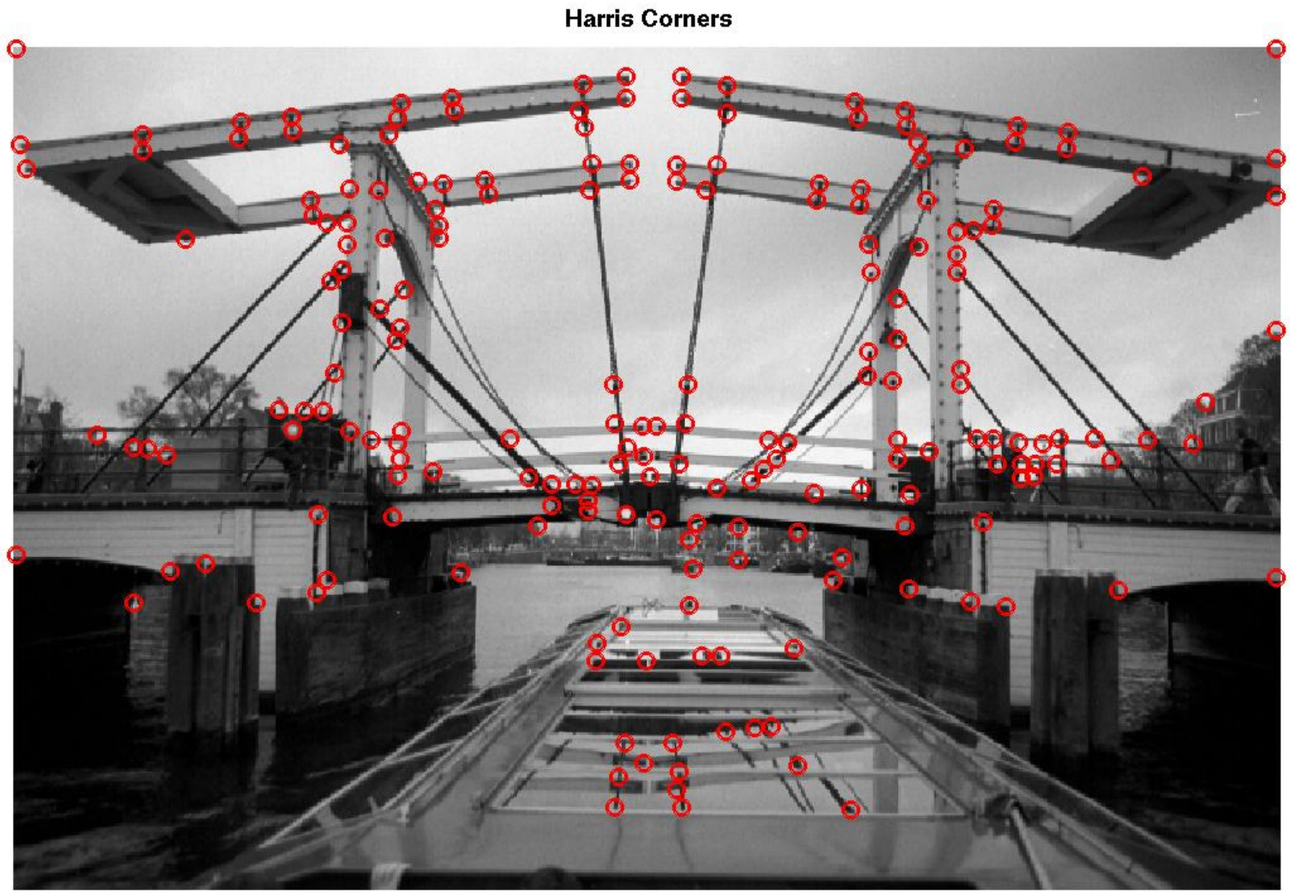




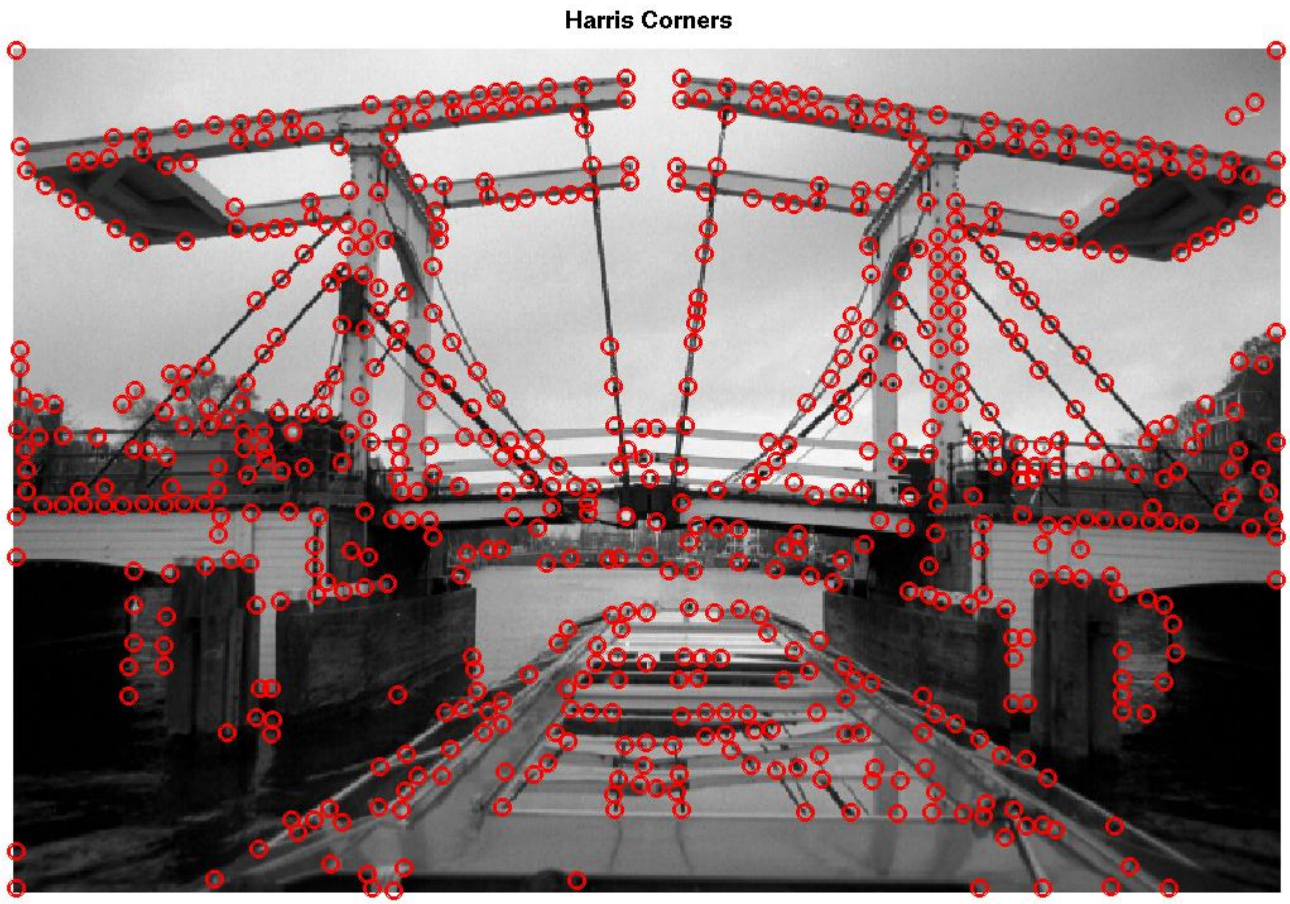
Plotting elipsses corresponding the the ‘corner’ matrix’
(changing the area over which statistics is averaged)

```
% Harris Corner detector - by Kashif Shahzad
sigma=2; thresh=0.1; sze=11; disp=0;

% Derivative masks
dy = [-1 0 1; -1 0 1; -1 0 1];
dx = dy'; %dx is the transpose matrix of dy

% Ix and Iy are the horizontal and vertical edges of image
Ix = conv2(bw, dx, 'same');
Iy = conv2(bw, dy, 'same');

% Calculating the gradient of the image Ix and Iy
g = fspecial('gaussian',max(1,fix(6*sigma)), sigma);
Ix2 = conv2(Ix.^2, g, 'same');                    % Smoothed squared image derivatives
Iy2 = conv2(Iy.^2, g, 'same');
Ixy = conv2(Ix.*Iy, g, 'same');

% My preferred  measure according to research paper
cornerness = (Ix2.*Iy2 - Ixy.^2)./(Ix2 + Iy2 + eps);

% We should perform nonmaximal suppression and threshold
mx = ordfilt2(cornerness,sze^2,ones(sze));              % Grey-scale dilate
cornerness = (cornerness==mx)&(cornerness>thresh); % Find maxima
[rws,cols] = find(cornerness);

clf ; imshow(bw); hold on;
p=[cols rws];
plot(p(:,1),p(:,2),'or'); title('\bf Harris Corners')
```

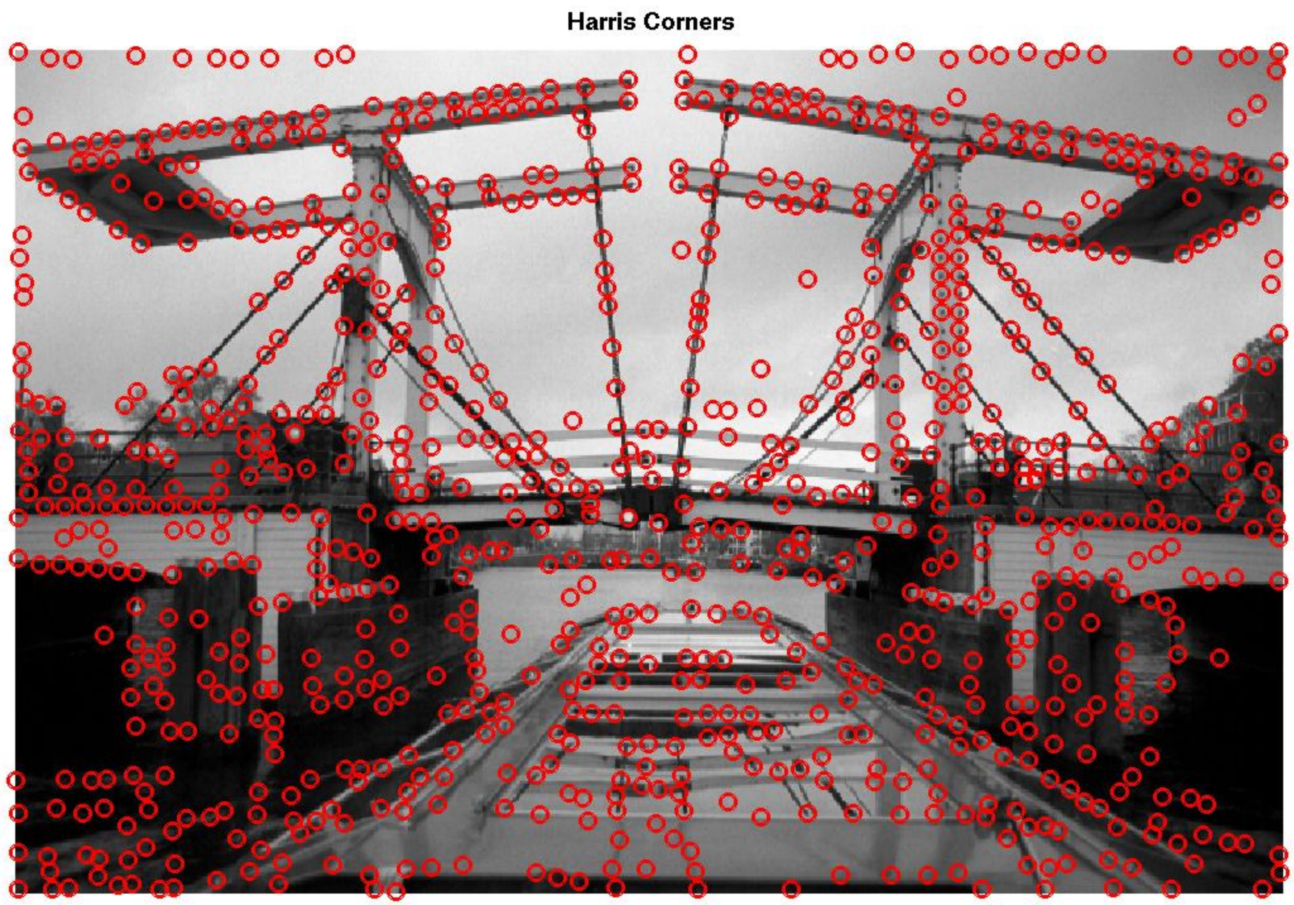
Example (σ=0.1)



Example (σ=0.01)

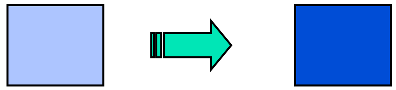


Example (σ=0.001)



Harris Corner Detector - Example

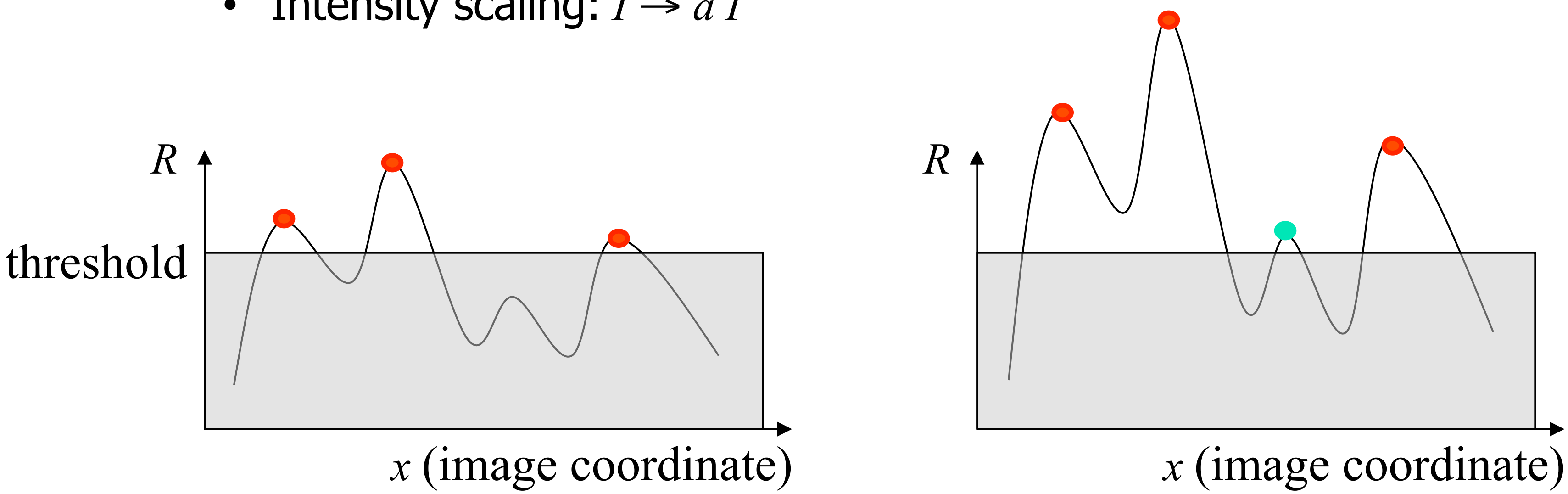
Affine intensity change



$$I \rightarrow a\,I + b$$

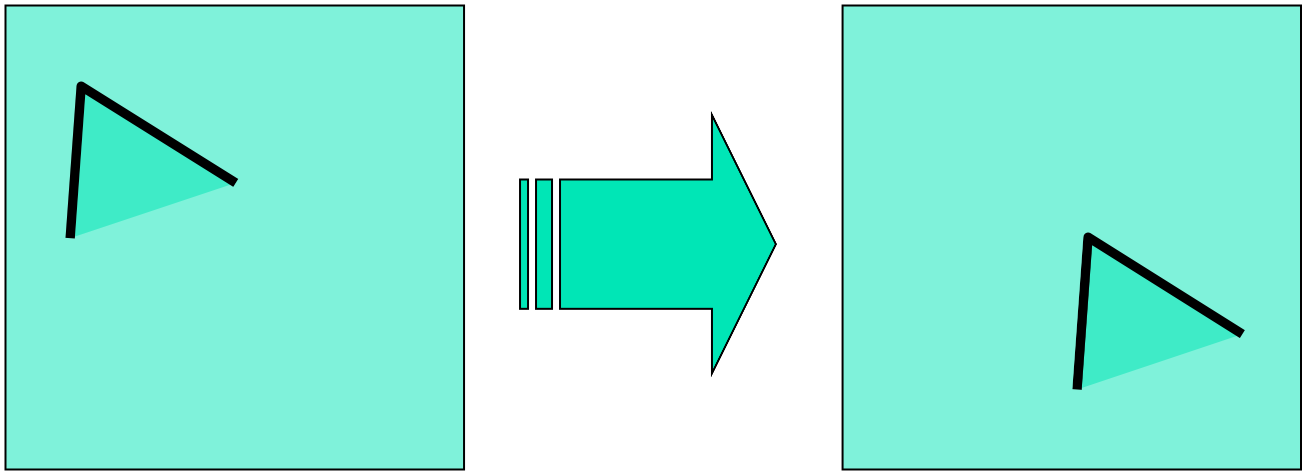
- Only derivatives are used => invariance to intensity shift $I \rightarrow I + b$

- Intensity scaling: $I \rightarrow a\,I$



Partially invariant to affine intensity change

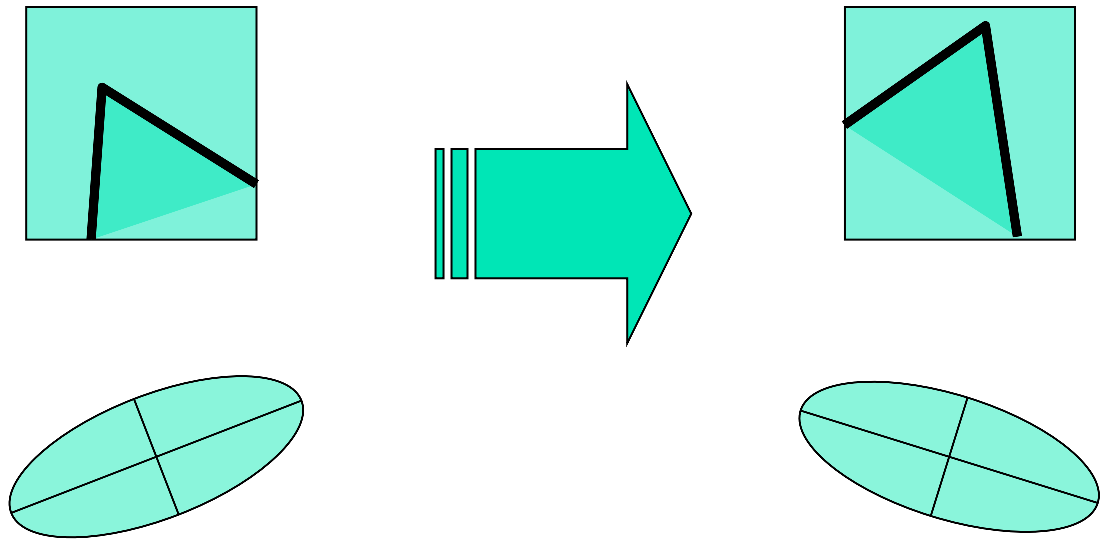
Image translation



- Derivatives and window function are shift-invariant

Corner location is covariant w.r.t. translation

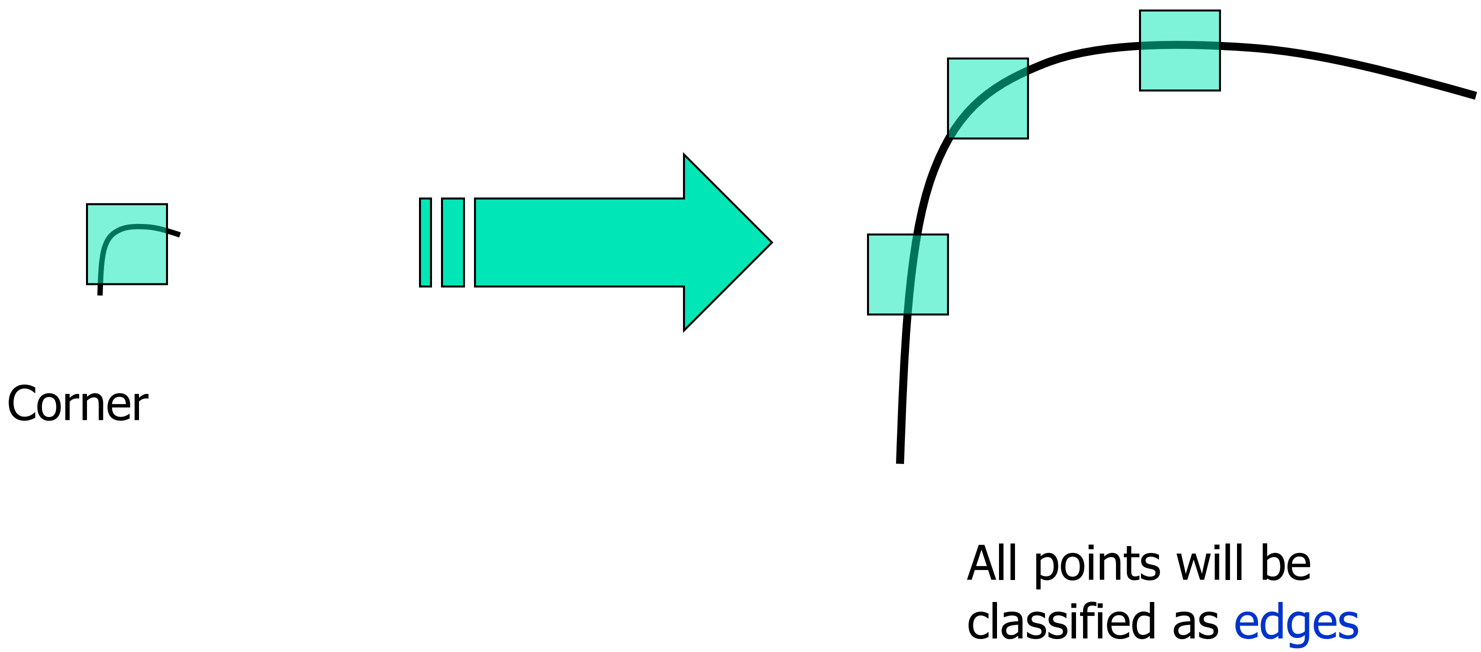
Image rotation



Second moment ellipse rotates but its shape (i.e. eigenvalues) remains the same

Corner location is covariant w.r.t. rotation

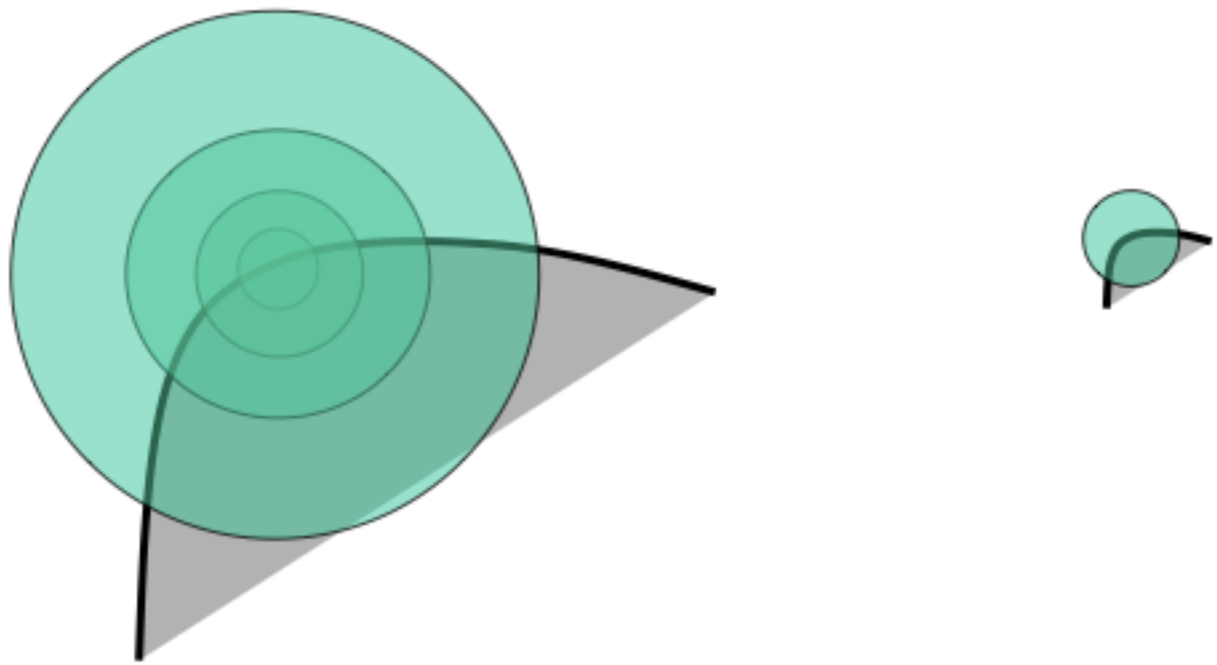
Scaling



Corner location is not covariant to scaling!

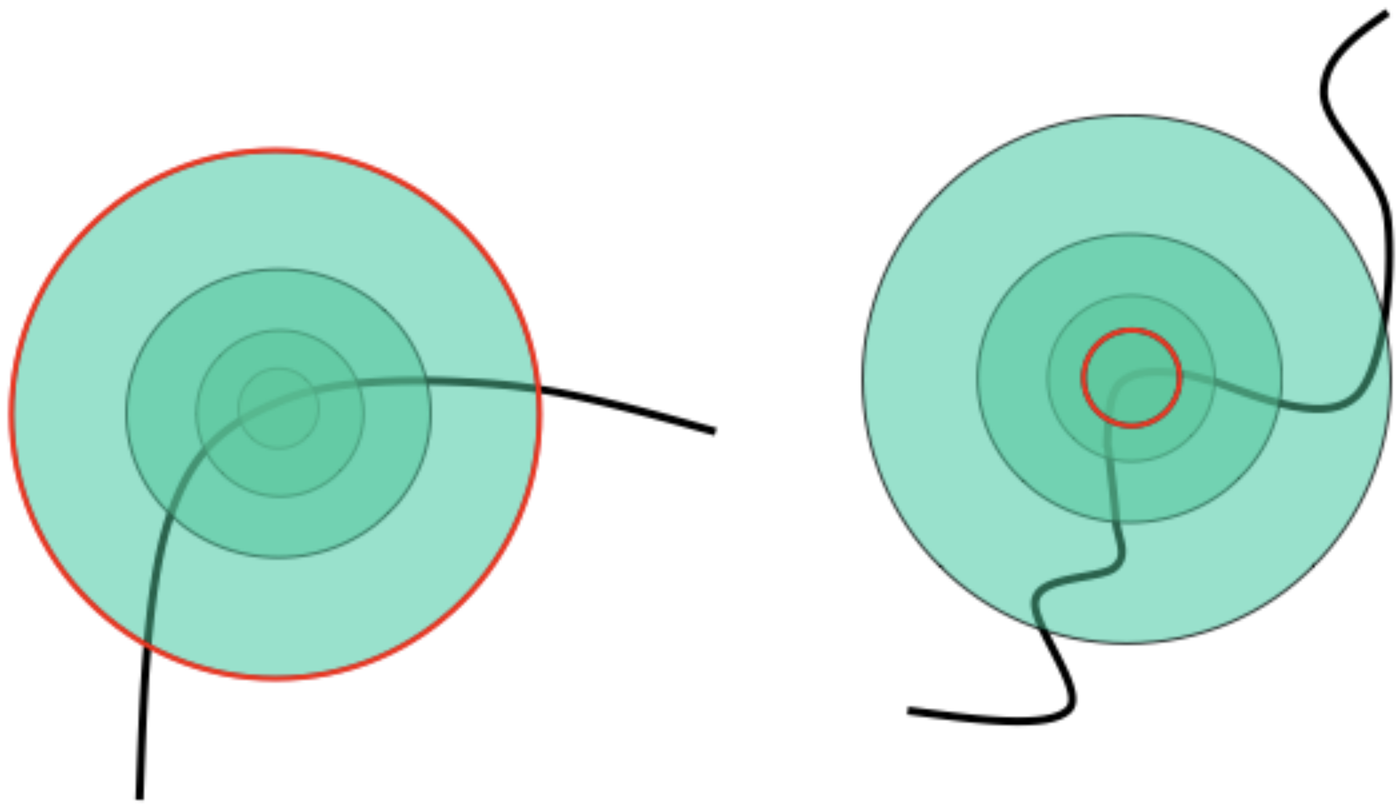
Scale Invariant Detection

- Consider regions of different size
- Select regions to subtend the same content



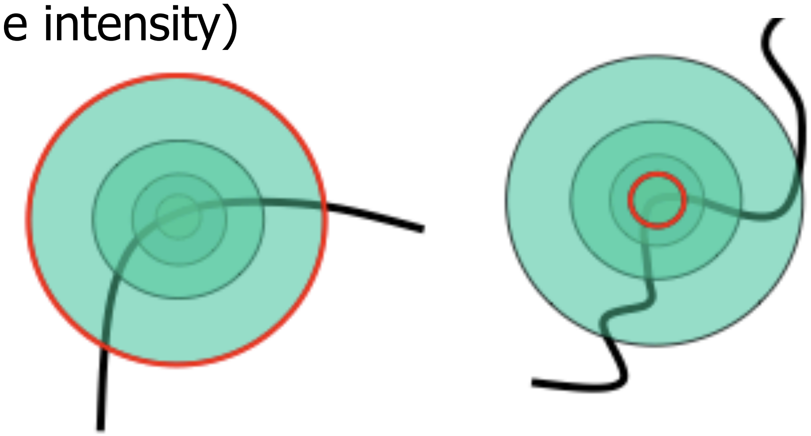
Scale Invariant detection

- How to choose the size of the region independently

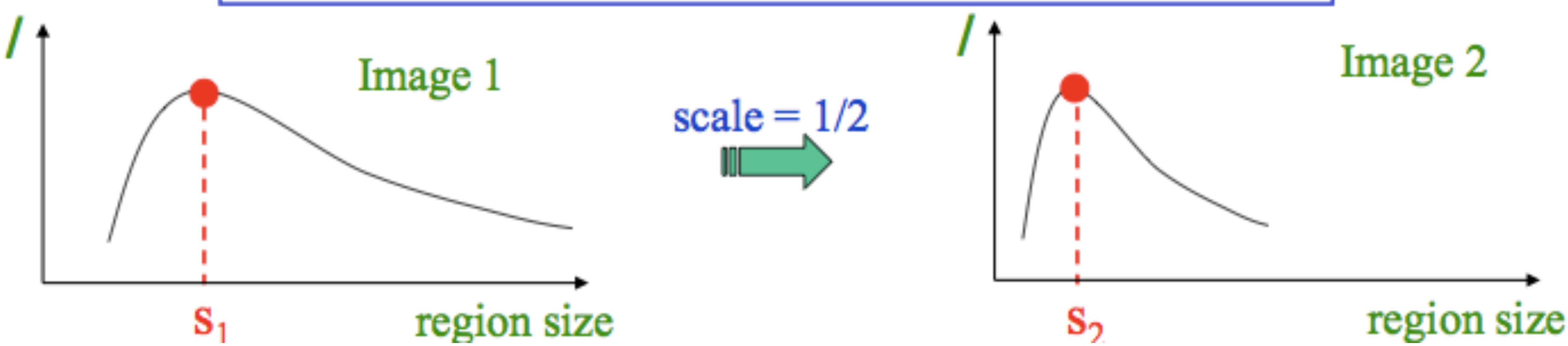


Scale invariant detection

Idea: design a function over region which remains constant as the size of the region Changes (e.g. average intensity)

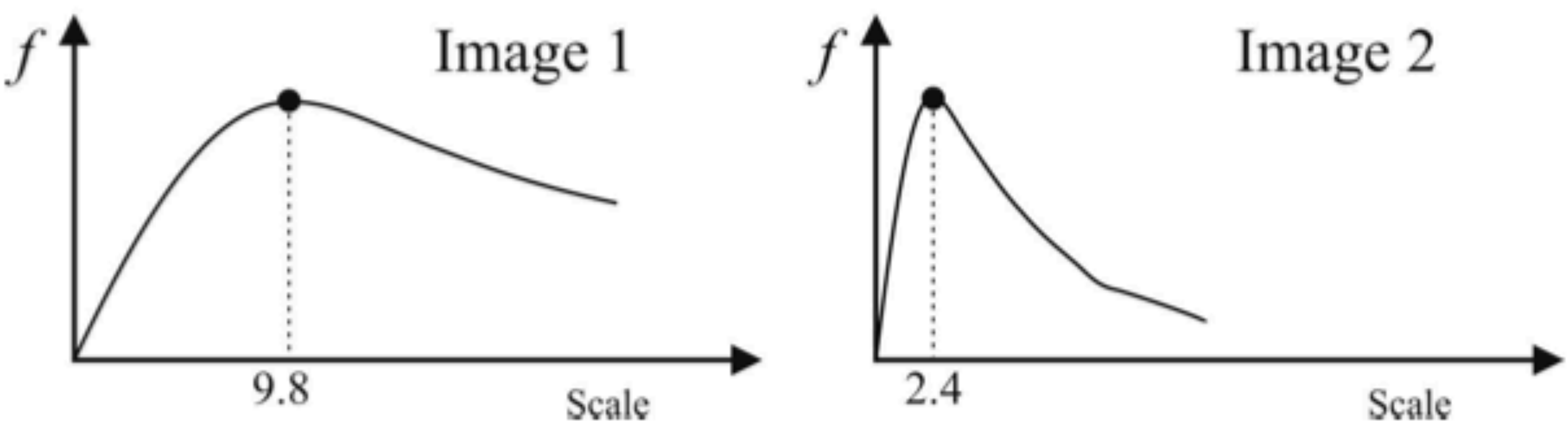


Important: this scale invariant region size is found in each image independently!



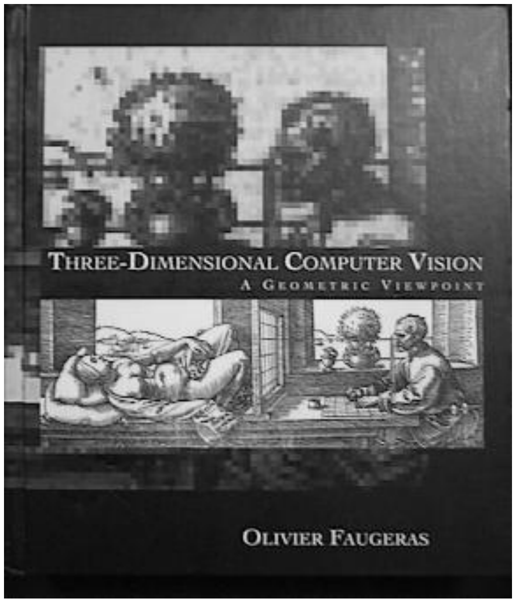
Scale Invariant detection

- Sharp local intensity changes are good functions for identifying relative scale of the region
- Response of Laplacian of Gaussians (LoG) at a point



Improved Invariance Handling

Want to find



… in here

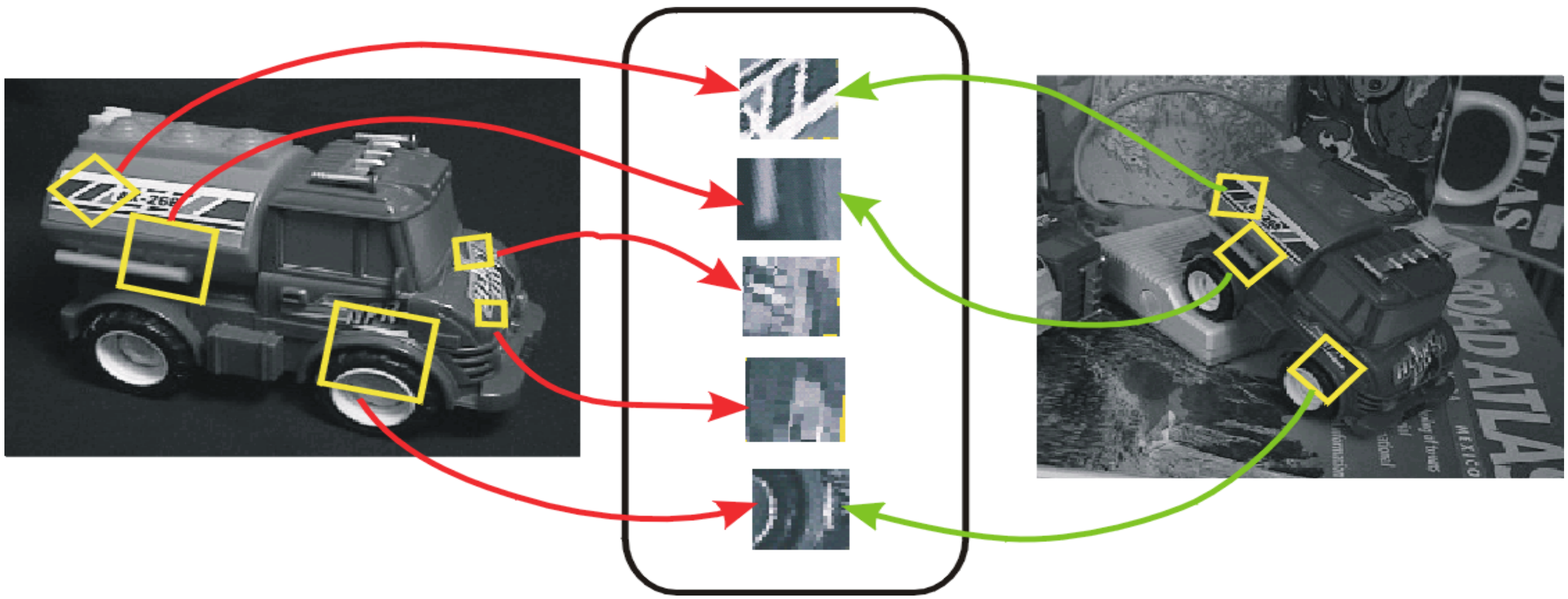
SIFT Reference

Distinctive image features from scale-invariant keypoints. David G. Lowe, International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.

SIFT = Scale Invariant Feature Transform

Invariant Local Features

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



SIFT Features

Advantages of invariant local features

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

SIFT On-A-Slide

1. Enforce invariance to scale: Compute Gaussian difference max, for many different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

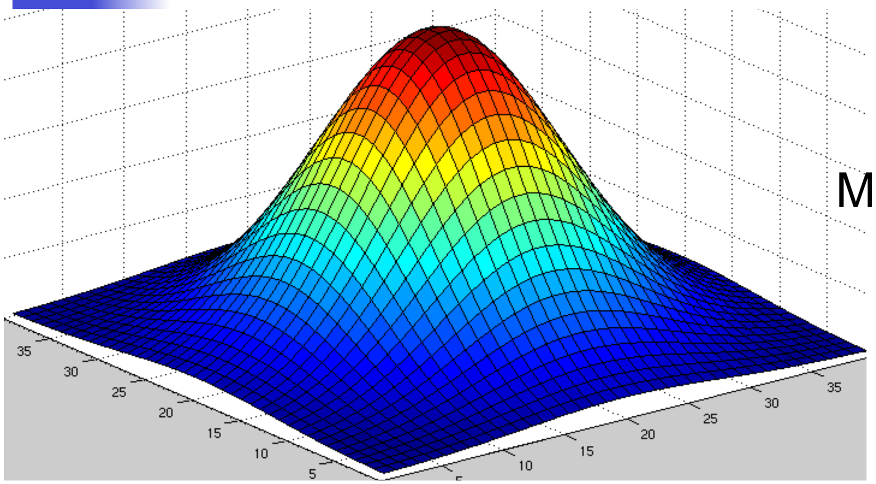
Finding "Keypoints" (Corners)

Idea: Find Corners, but scale invariance

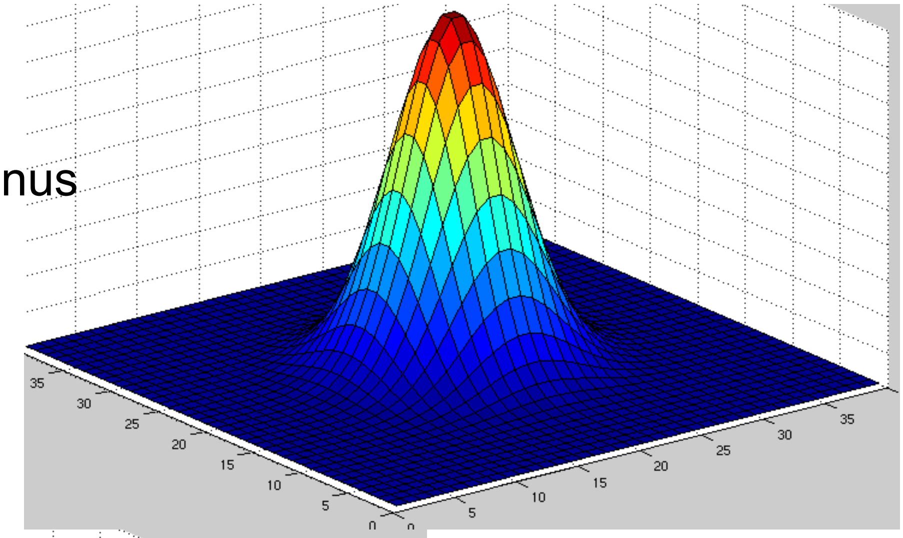
Approach:

- Run linear filter (difference of Gaussians)
- Do this at different resolutions of image pyramid

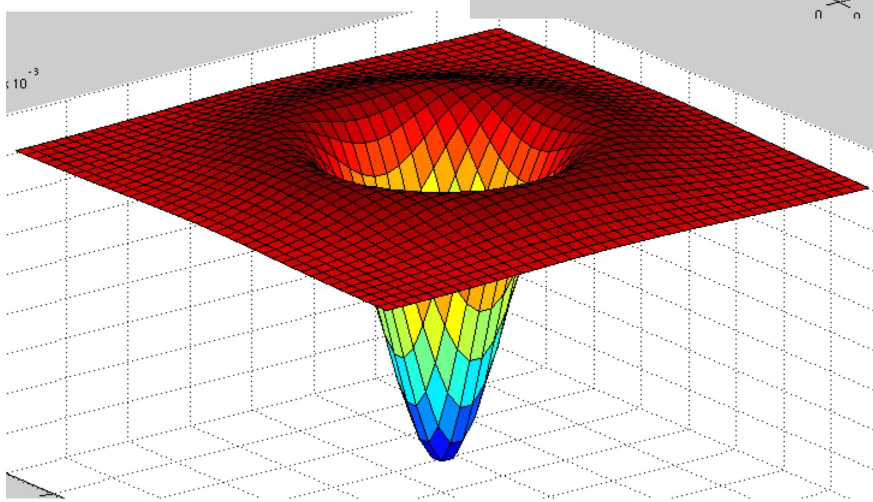
Difference of Gaussians



Minus



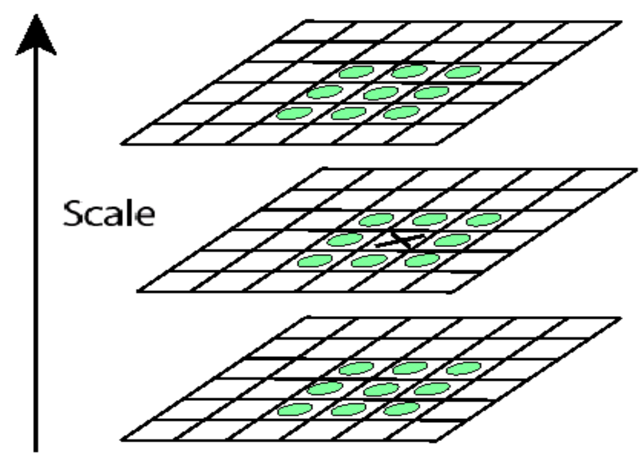
Equals



Approximates Laplacian (see filtering lecture)

Key point localization

- In D. Lowe's paper image is decomposed to octaves (consecutively sub-sampled versions of the same image)
- Instead of convolving with large kernels within an octave kernels are kept the same
- Detect maxima and minima of difference-of-Gaussian in scale space
- Look for 3x3 neighbourhood in scale and space



Example of keypoint detection

(a)

(b)

(a) 233x189 image
(b) 832 DOG extrema
(c) 729 above threshold

SIFT On-A-Slide

1. Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

Example of keypoint detection

Threshold on value at DOG peak and on ratio of principle curvatures (Harris approach)


(c)


(d)

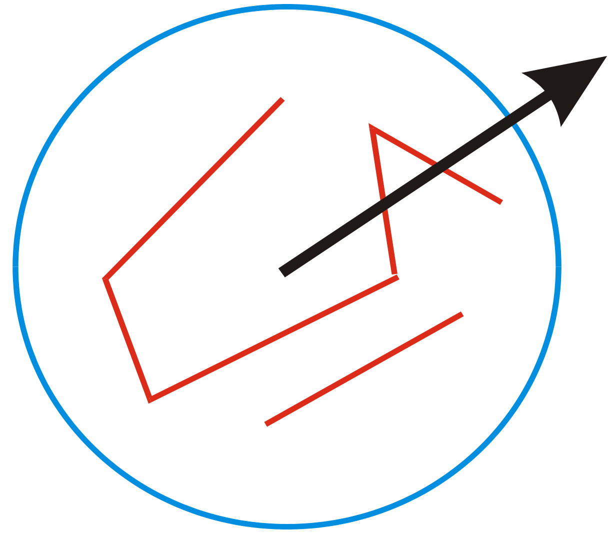
(c) 729 left after peak value threshold (from 832)
(d) 536 left after testing ratio of principle curvatures

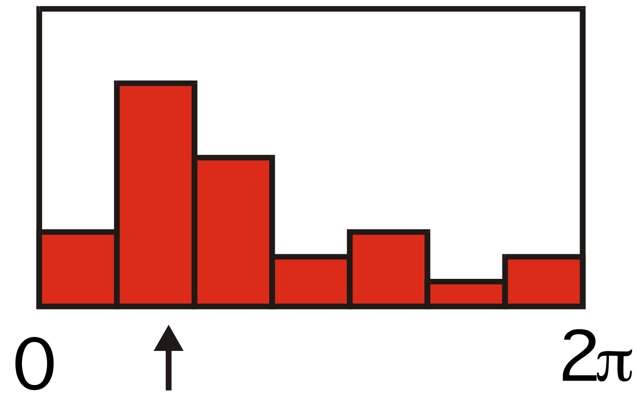
SIFT On-A-Slide

1. Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

Select canonical orientation

- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each key specifies stable 2D coordinates (x, y, scale, orientation)



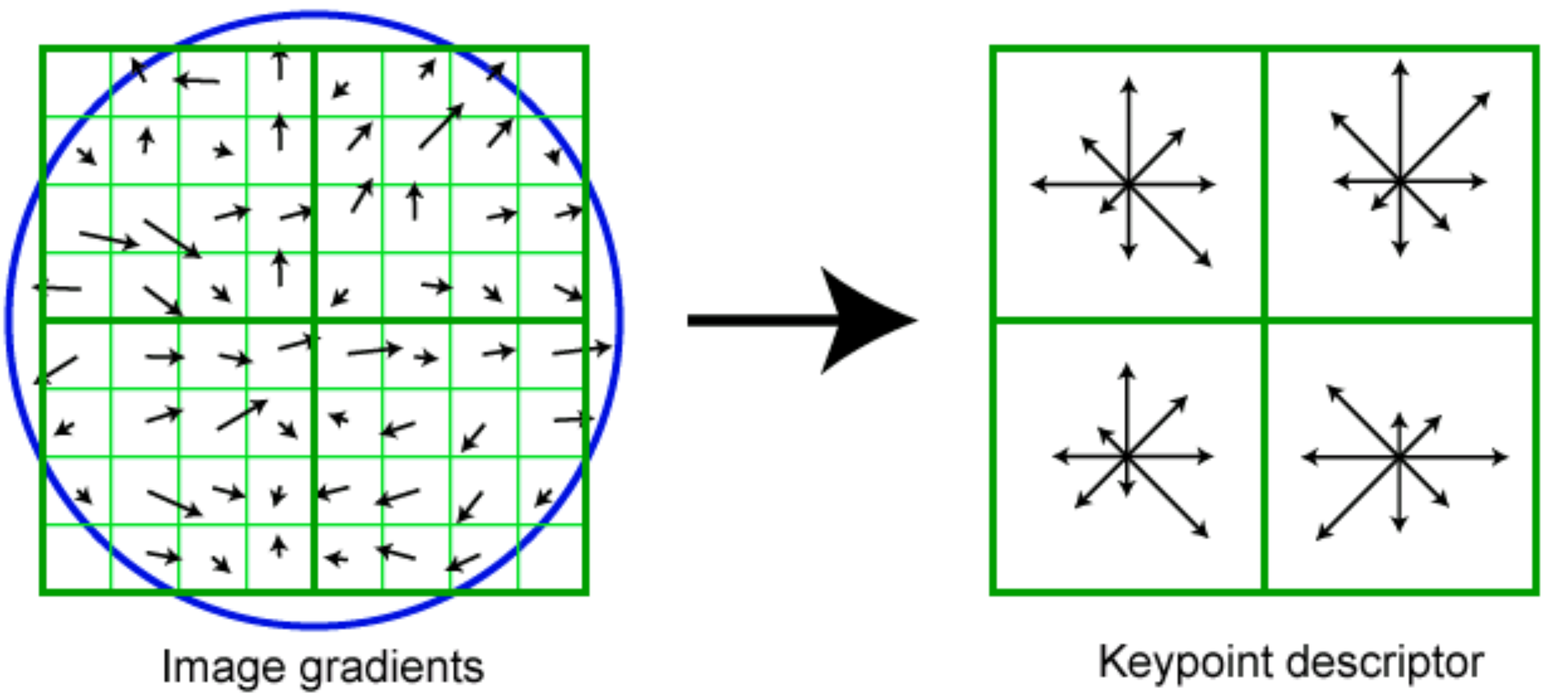


SIFT On-A-Slide

1. Enforce invariance to scale: Compute Gaussian difference max, for may different scales; non-maximum suppression, find local maxima: keypoint candidates
2. Localizable corner: For each maximum fit quadratic function. Compute center with sub-pixel accuracy by setting first derivative to zero.
3. Eliminate edges: Compute ratio of eigenvalues, drop keypoints for which this ratio is larger than a threshold.
4. Enforce invariance to orientation: Compute orientation, to achieve rotation invariance, by finding the strongest second derivative direction in the smoothed image (possibly multiple orientations). Rotate patch so that orientation points up.
5. Compute feature signature: Compute a "gradient histogram" of the local image region in a 4x4 pixel region. Do this for 4x4 regions of that size. Orient so that largest gradient points up (possibly multiple solutions). Result: feature vector with 128 values (15 fields, 8 gradients).
6. Enforce invariance to illumination change and camera saturation: Normalize to unit length to increase invariance to illumination. Then threshold all gradients, to become invariant to camera saturation.

SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions



Nearest-neighbor matching to feature database

- Hypotheses are generated by **approximate nearest neighbor** matching of each feature to vectors in the database
  - SIFT use best-bin-first (Beis & Lowe, 97) modification to k-d tree algorithm
  - Use heap data structure to identify bins in order by their distance from query point

- **Result:** Can give speedup by factor of 1000 while finding nearest neighbor (of interest) 95% of the time

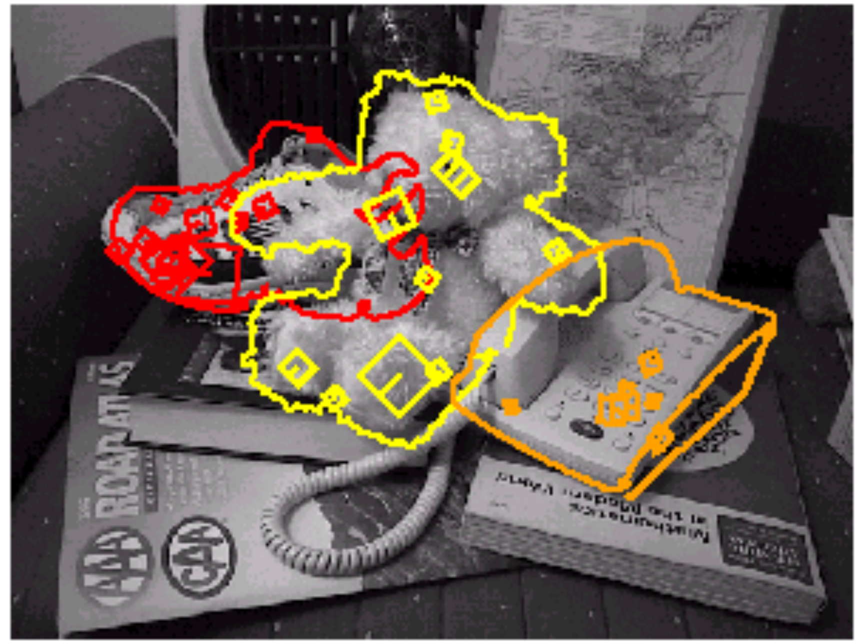
3D Object Recognition

- Extract outlines with background subtraction

3D Object Recognition

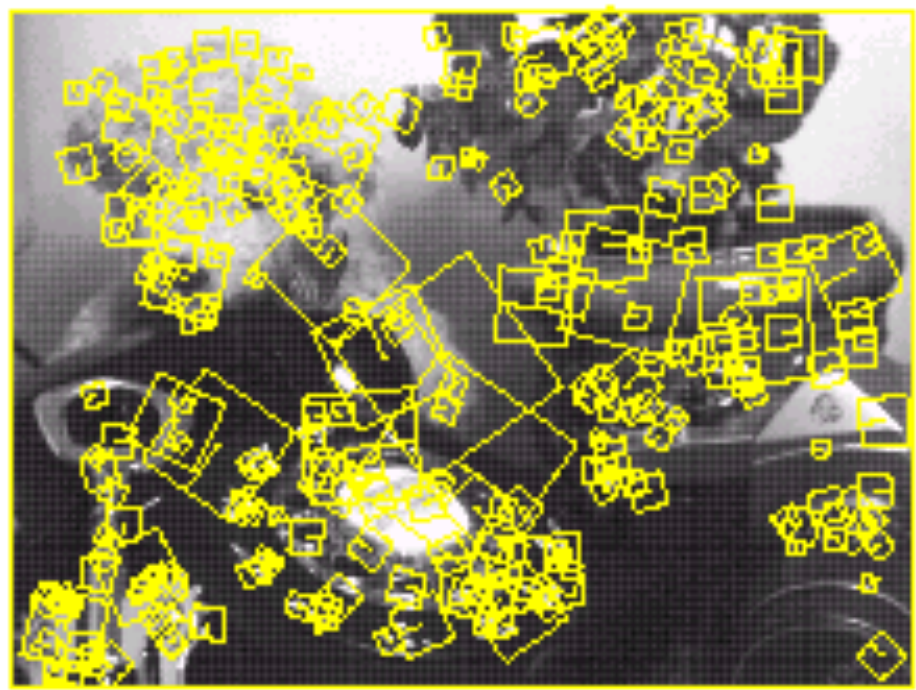
- Only 3 keys are needed for recognition, so extra keys provide robustness
- Affine model is no longer as accurate

Recognition under occlusion



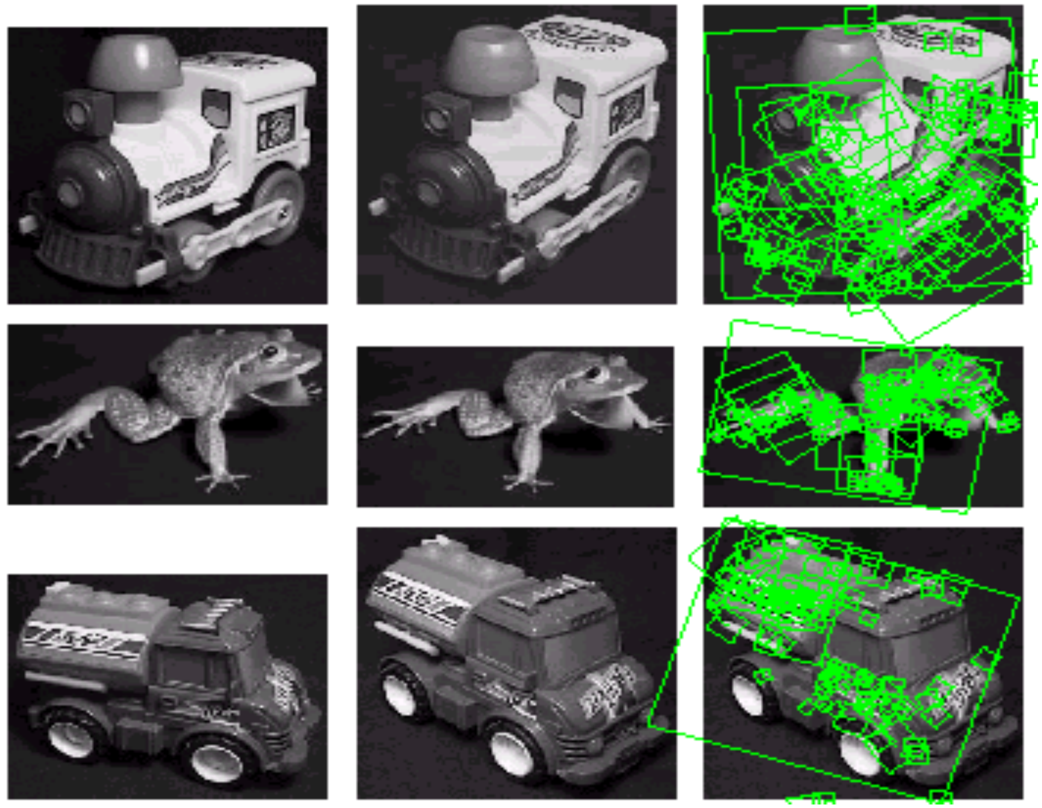
Test of illumination invariance

- Same image under differing illumination

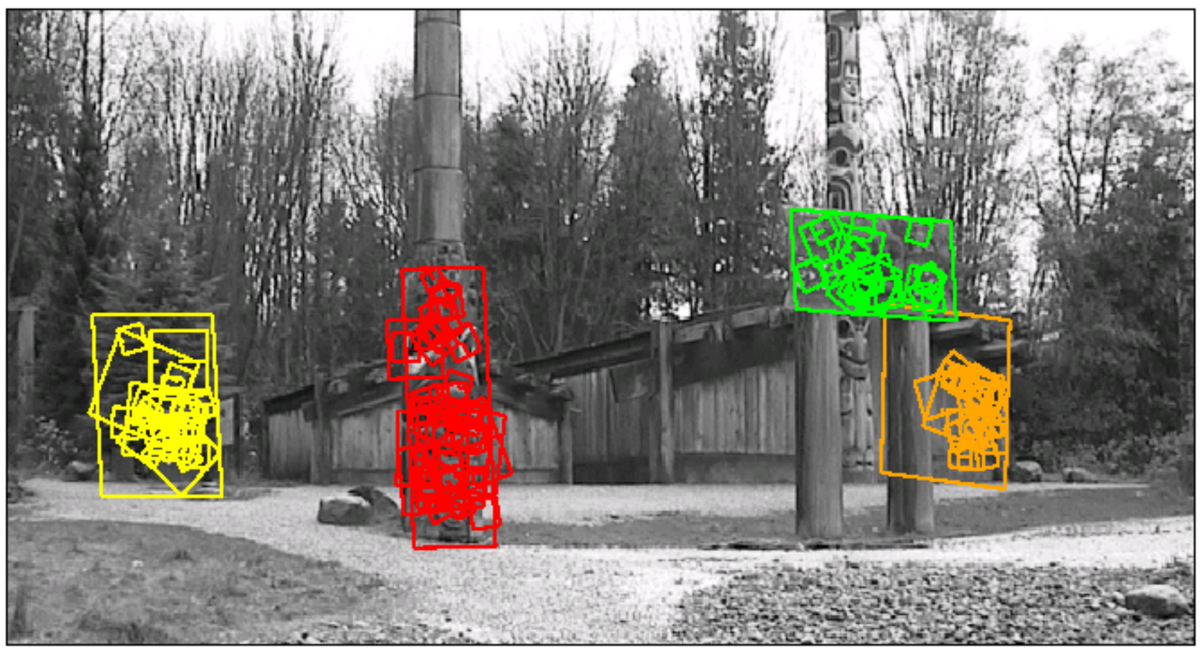


273 keys verified in final match

Examples of view interpolation



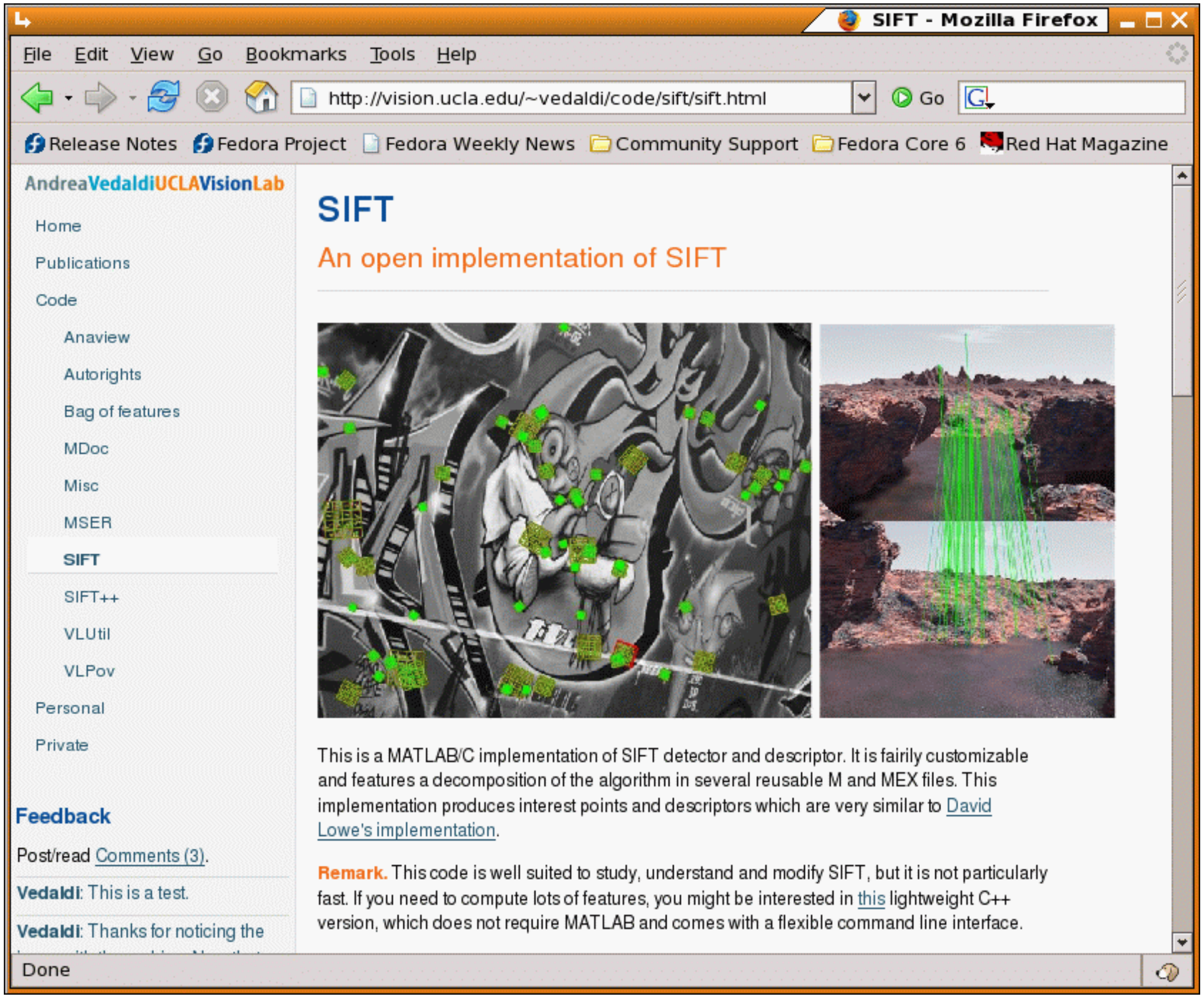
Location recognition



SIFT

- Invariances:
  - Scaling Yes
  - Rotation Yes
  - Illumination Yes
  - Perspective Projection Maybe
- Provides
  - Good localization Yes

SOFTWARE for Matlab (at UCLA, Oxford) www.VLFeat.org

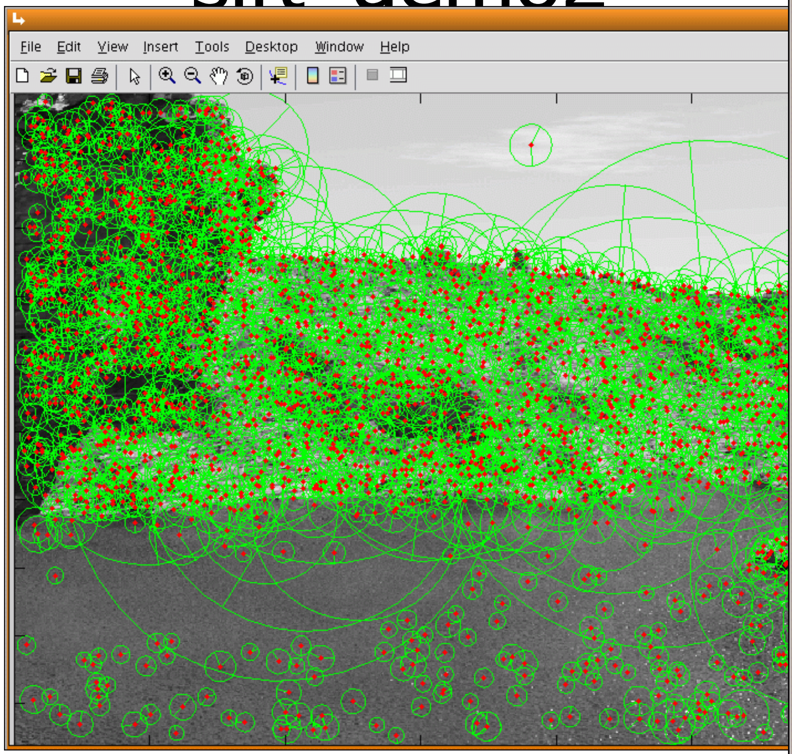


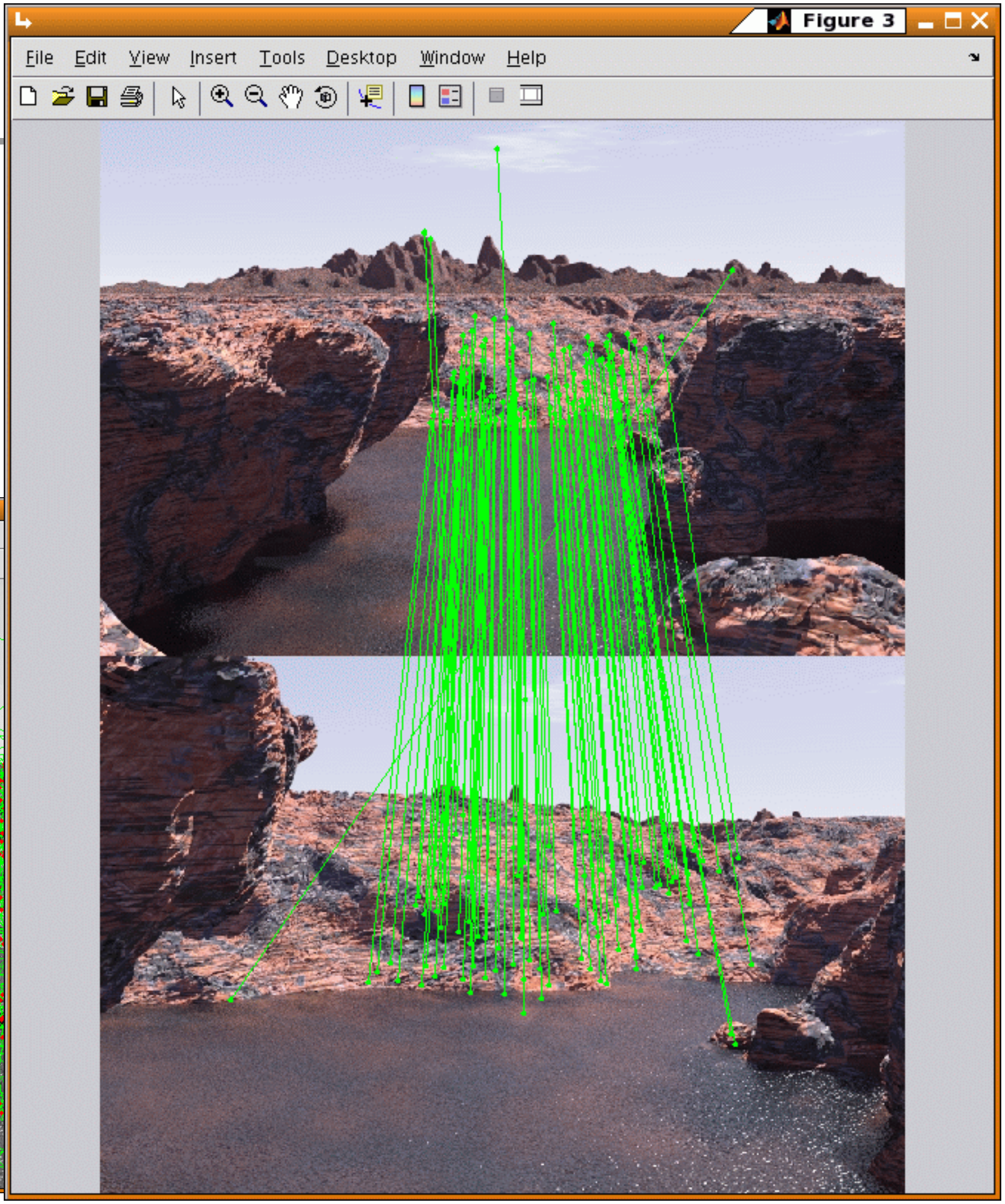
SIFT demos

Run

sift_compile

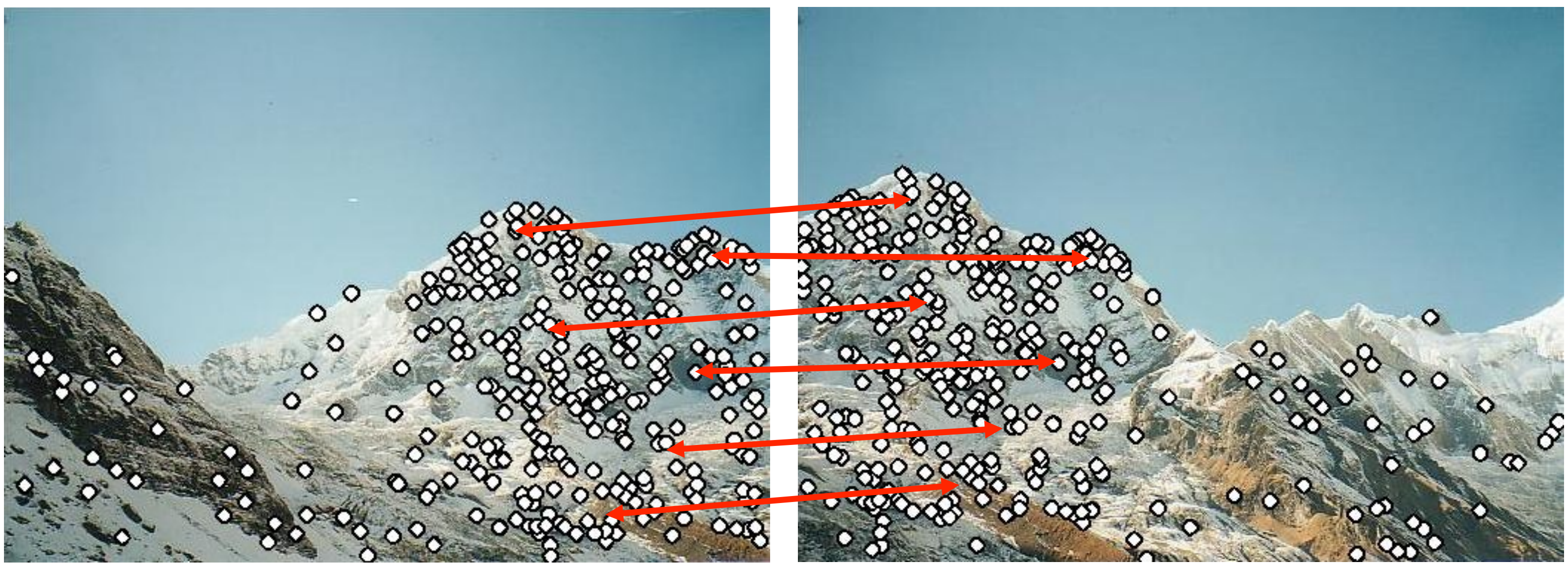
sift_demo2





Why extract features?

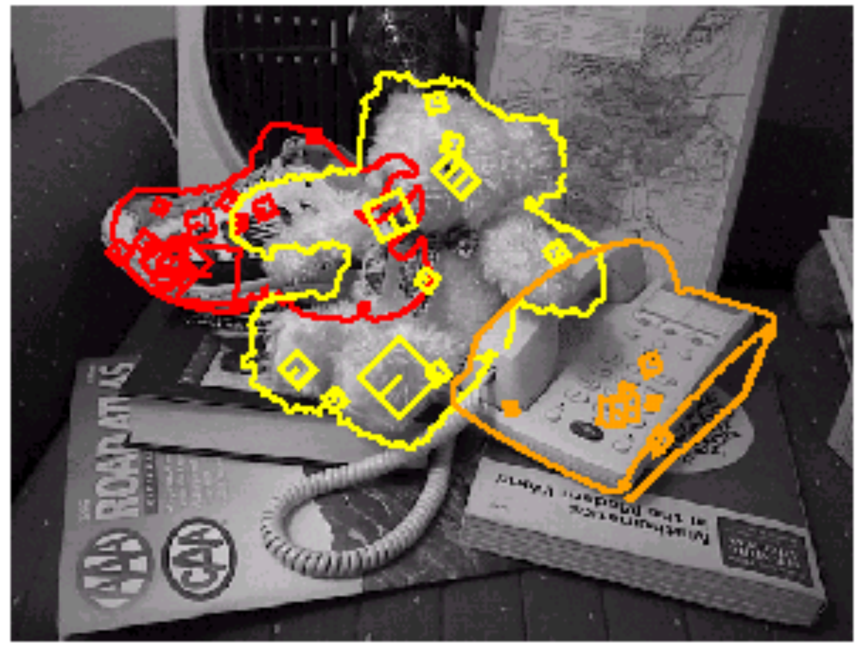
- Motivation: panorama stitching
  - We have two images – how do we combine them?



Step 1: extract features

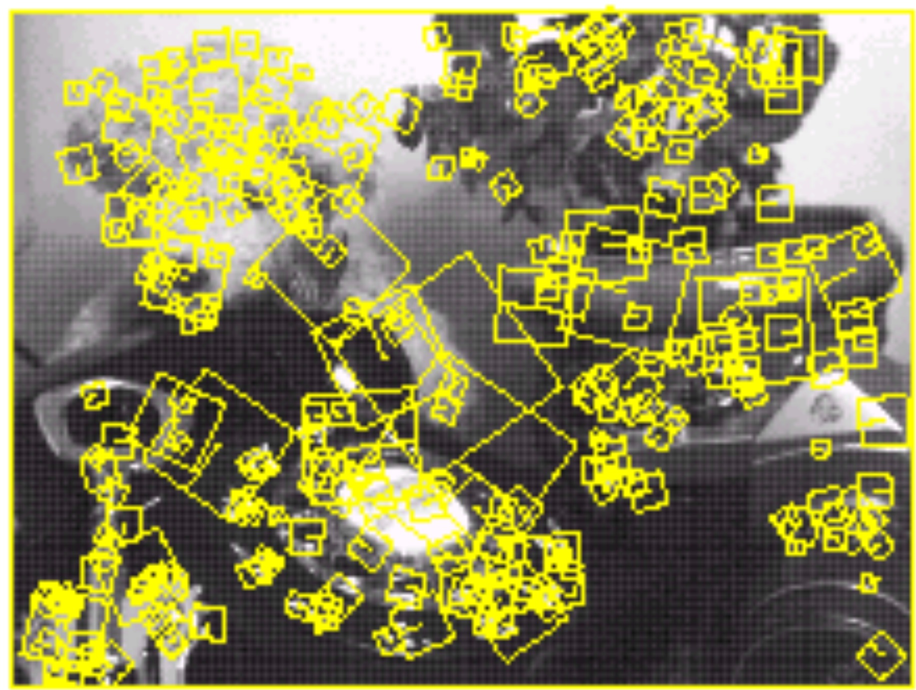
Step 2: match features

Recognition under occlusion



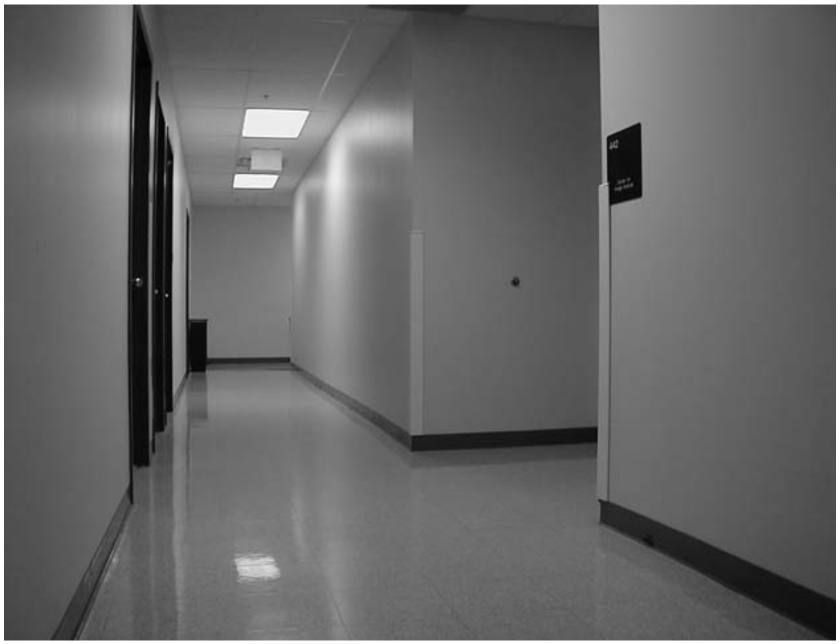
Test of illumination invariance

- Same image under differing illumination



273 keys verified in final match

Edge Detection



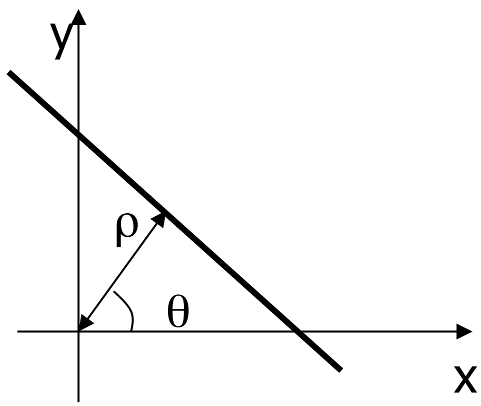
original image

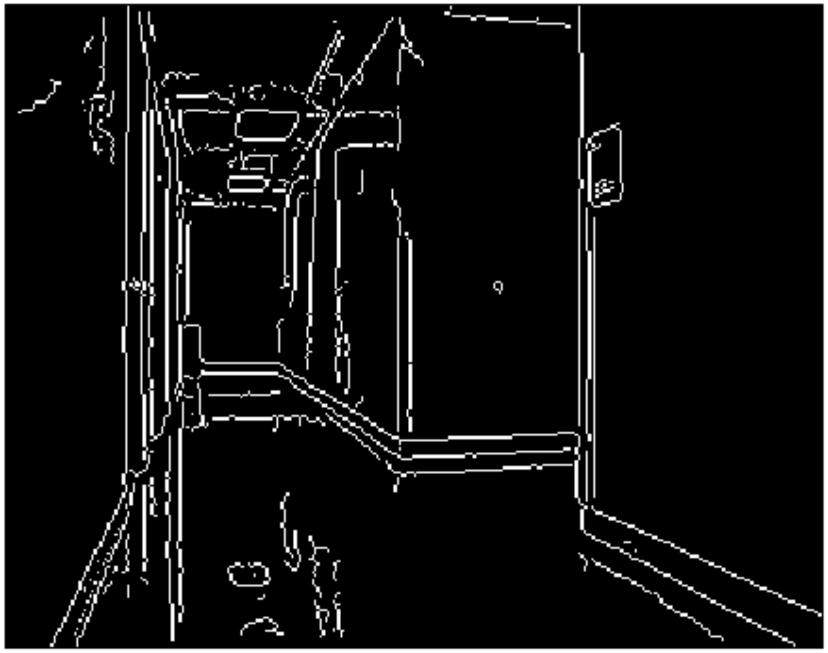
gradient magnitude

Canny edge detector

- Compute image derivatives
- if gradient magnitude > τ and the value is a local maximum along gradient direction – pixel is an edge candidate

Line fitting





Non-max suppressed gradient magnitude

- Edge detection, non-maximum suppression
  (traditionally Hough Transform – issues of resolution, threshold selection and search for peaks in Hough space)
- Connected components on edge pixels with similar orientation
  - group pixels with common orientation

Line fitting

$$A = \begin{bmatrix} \sum x_i^2 & \sum x_i y_i \\ \sum x_i y_i & \sum y_i^2 \end{bmatrix}$$

second moment matrix associated with each connected component
v_1 - eigenvector of A

$$v_1 = [\cos(\theta), \sin(\theta)]^T$$

$$\theta = \arctan(v_1(2)/v_1(1))$$

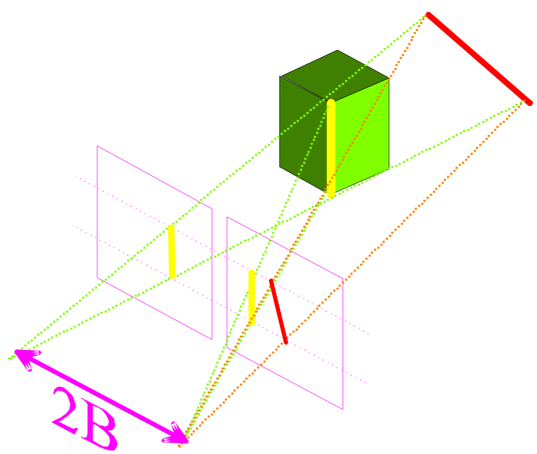
$$\rho = \bar{x}\sin(\theta) - \bar{y}\cos(\theta)$$

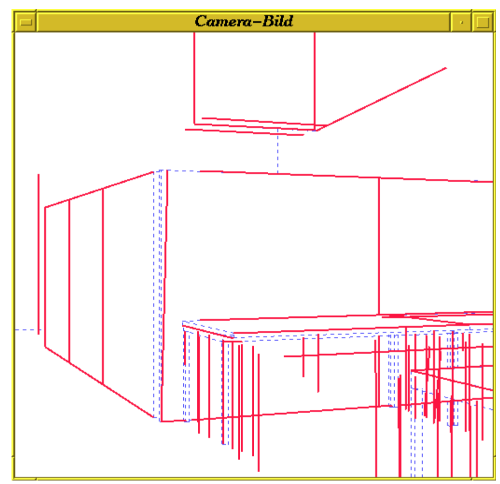
- Line fitting lines determined from eigenvalues and eigenvectors of A
- Candidate line segments - associated line quality

Stereo Feature Based Reconstruction

Correspondence Problem:

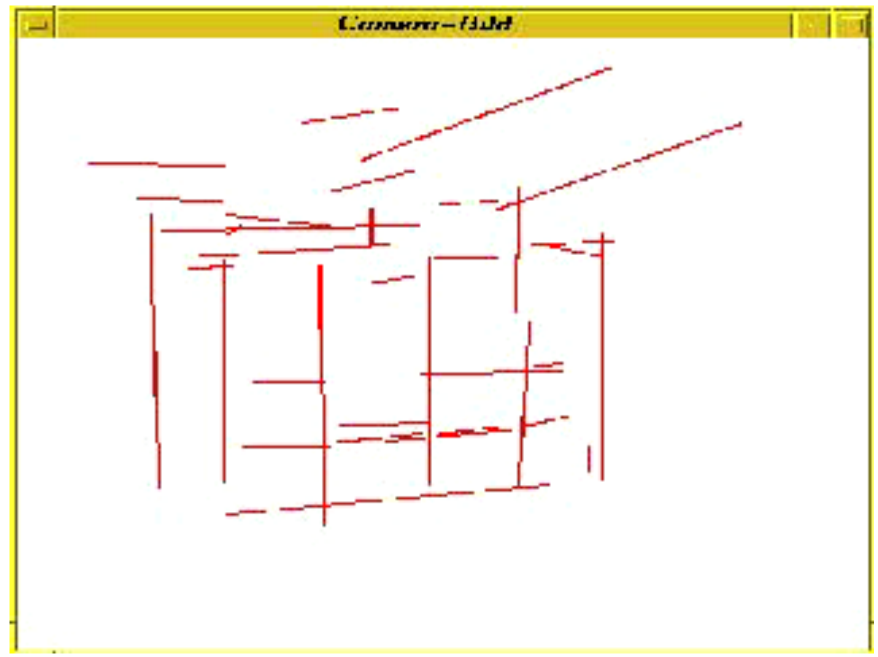
• How to find corresponding areas of two camera images (points, line segments, curves, regions)





• In feature-based matching, the idea is to pick a feature type (e.g. edges), define a matching criteria (e.g. orientation and contrast sign), and then look for matches within a disparity range
• Feature Matching later

Results - Reconstruction



Color Vision

- With the advent of inexpensive color imagery and processing, color information can be used effectively for machine vision.
- Color provides multiple information per pixel, often enabling complex classification.
- Perception of Color depends on three factors:
  - The spectrum of energy in various wavelengths illuminating the object surface,
  - The spectral reflectance of the object surface, which determines how the surface changes the received spectrum into the radiated spectrum,
  - The spectral sensitivity of the sensor irradiated by the object's surface.

Color Image

Full color

Red

Green

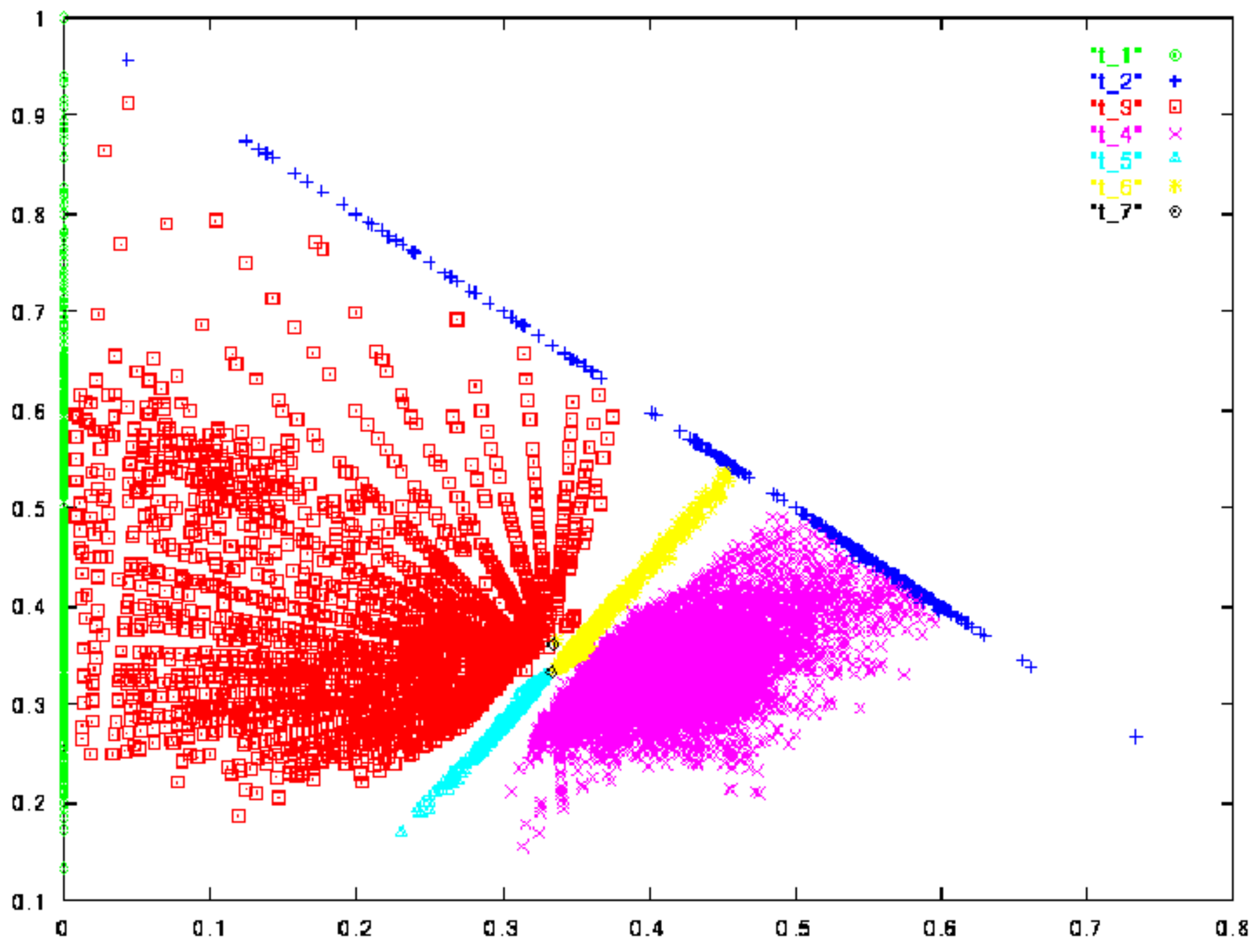
Blue

Jana Kosecka

Application

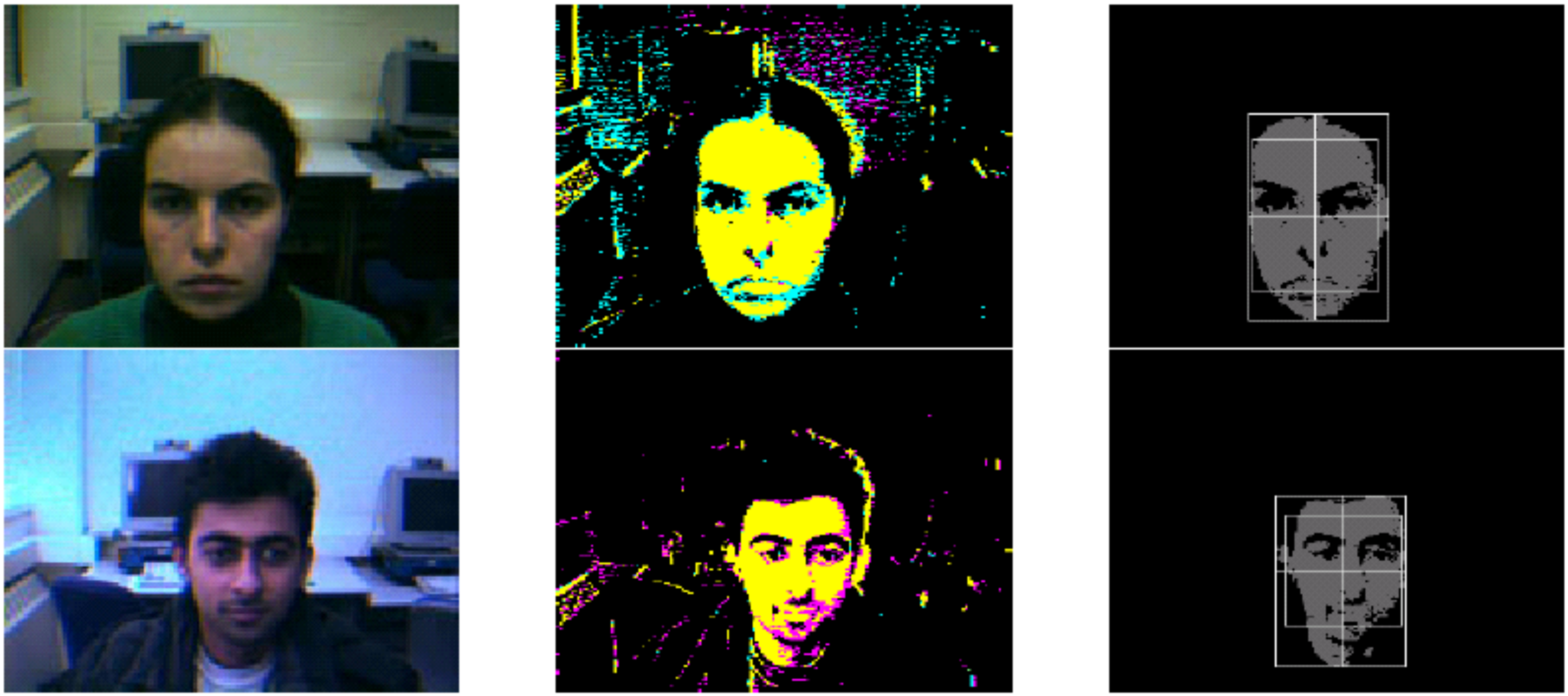
- Color Consider the problem of locating/segmenting faces from images using color.
- First we need to identify the range of colors that could be associated with a face.
- The lighting conditions would play a significant role.
- Even under uniform illumination, other objects could fall into that color space. In this case we could use shape information for the purpose of segmentation.

Color space analysis



T4 – primary face color, t-5 and t-6 secondary face clusters

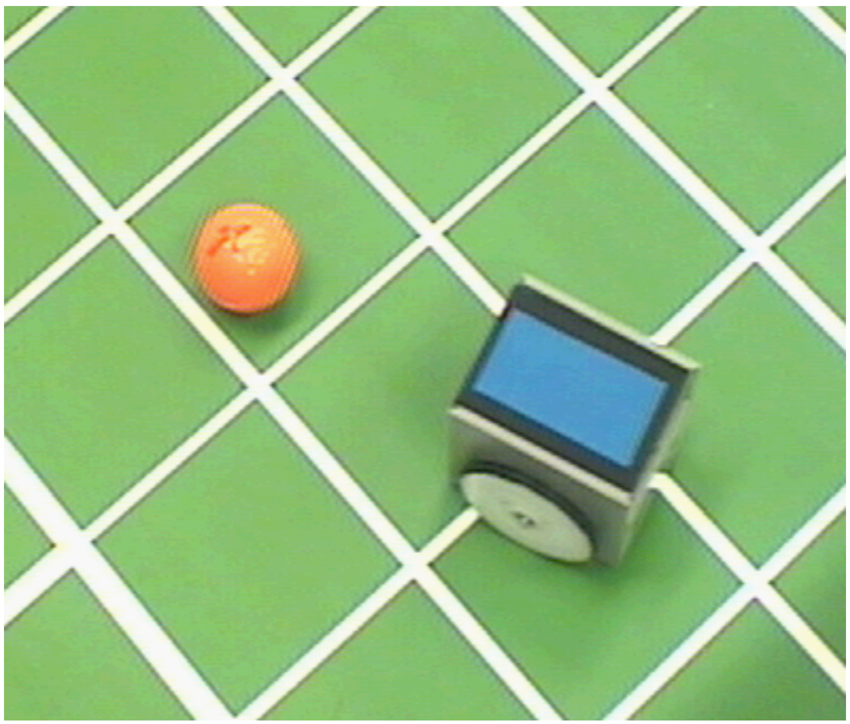
- Three major steps are involved in the face segmentation procedure
- First we need to create a labeled image based on the training data for identifying the color space that would represent the face.
- Connected component is used to merge regions that would be part of the face.
- The face is identified as the largest component and areas close to the components are merged.

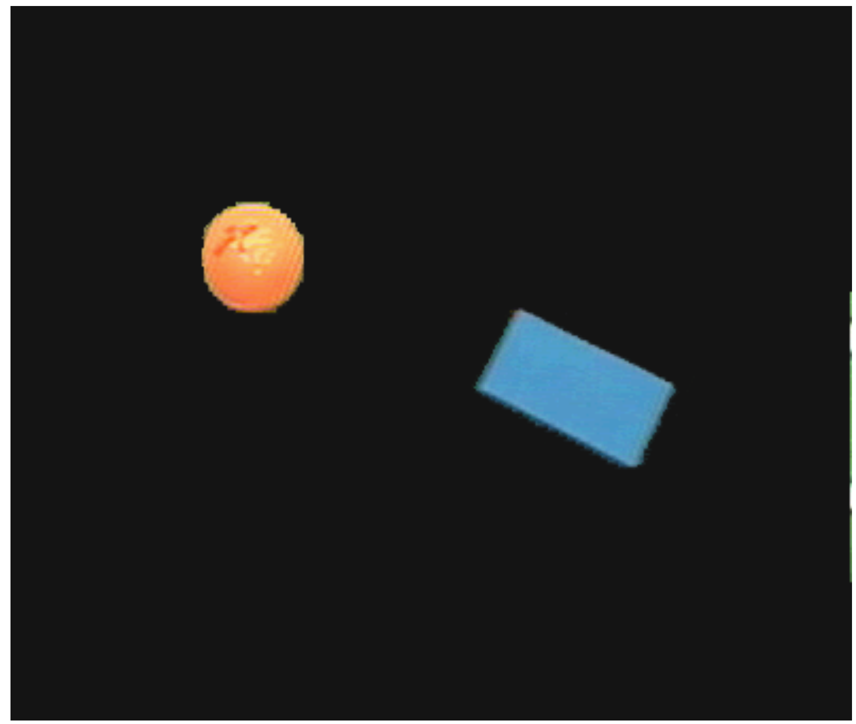


Jana Kosecka

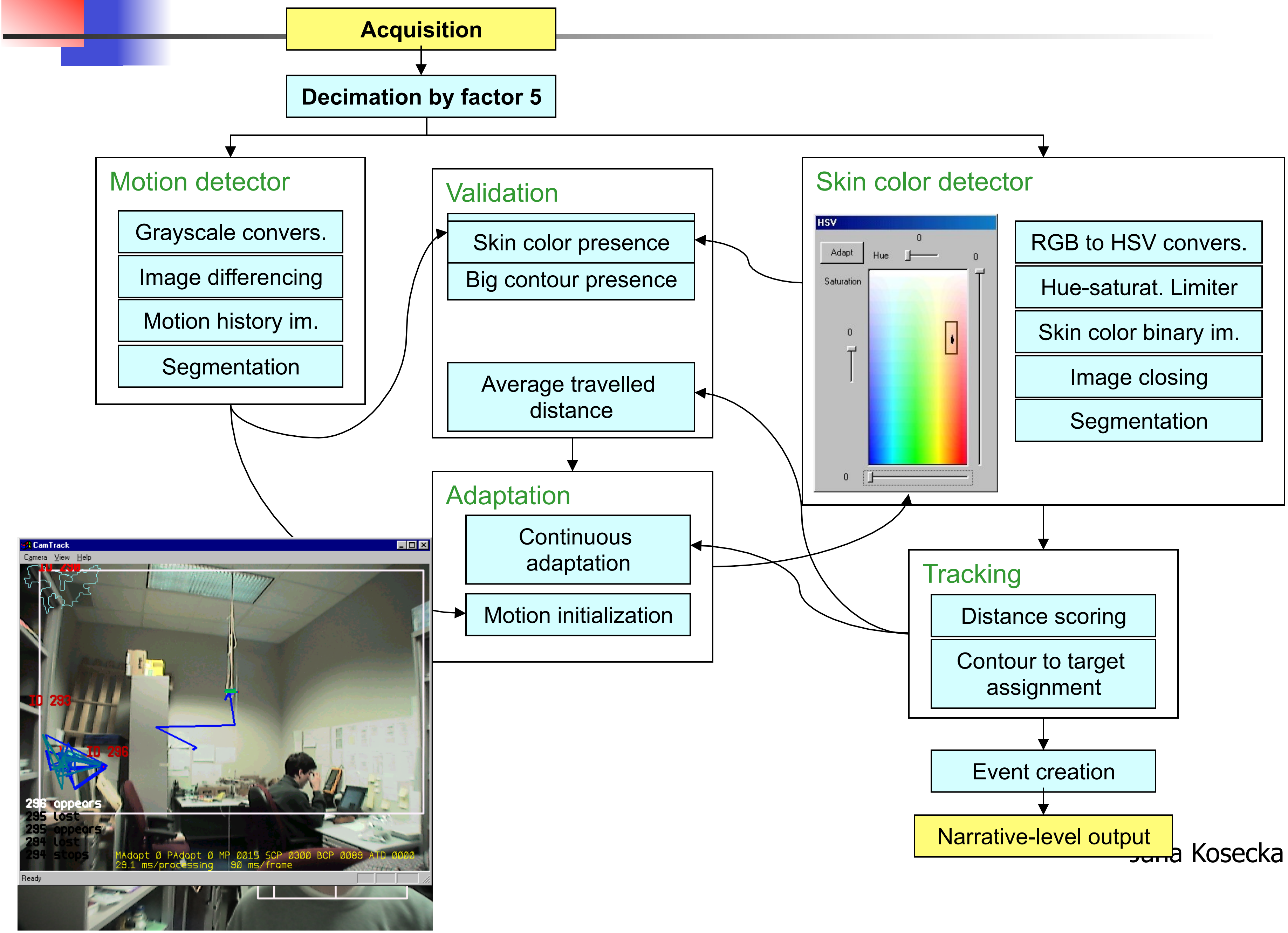
Color Tracking Sensors

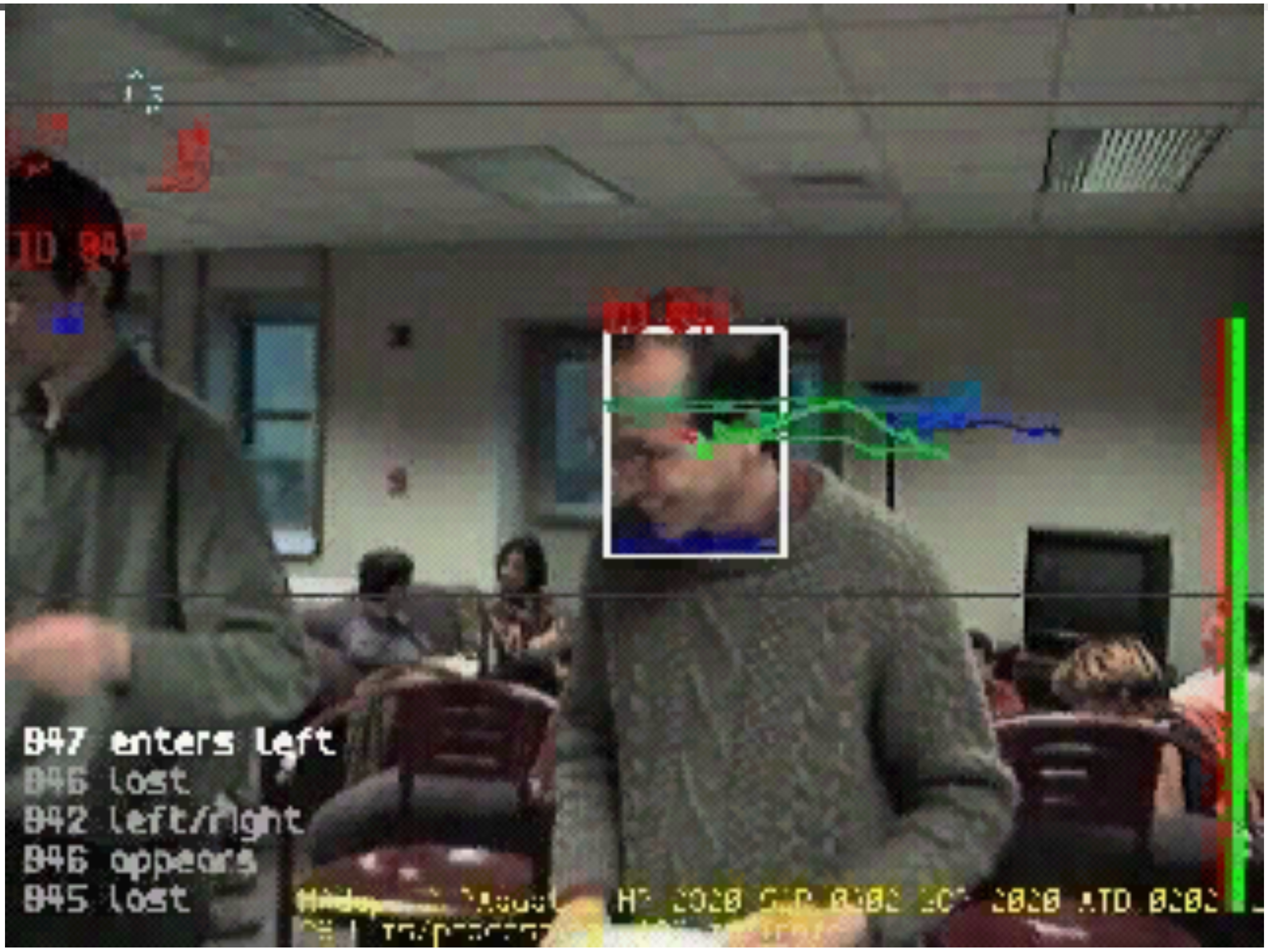
- Motion estimation of ball and robot for soccer playing using color tracking





Adaptive Human-Motion Tracking





Jana Kosecka

Image Primitives and Correspondence

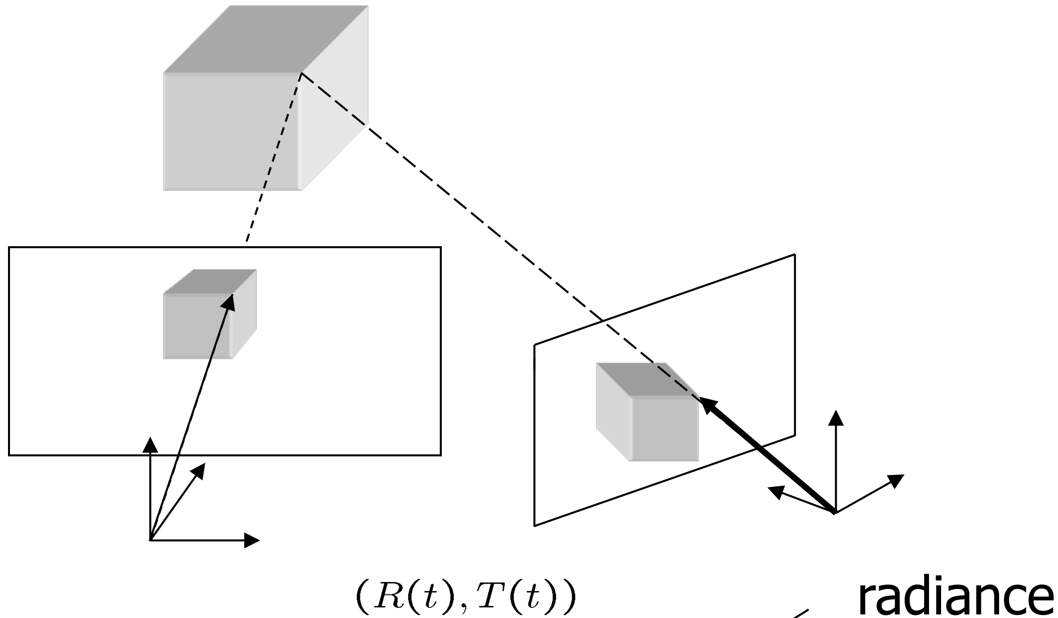


Given an image point in left image, what is the (corresponding) point in the right image, which is the projection of the same 3-D point

Image Primitives and Correspondence

Difficulties – ambiguities, large changes of appearance, due to change
Of viewpoint, non-uniquess

Matching - Correspondence



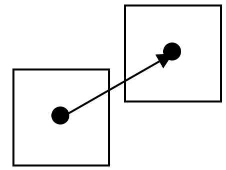
Lambertian assumption $\quad I_1(\mathbf{x}_1) = \mathcal{R}(p) = I_2(\mathbf{x}_2)$

Rigid body motion $\quad \mathbf{x}_2 = h(\mathbf{x}_1) = \frac{1}{\lambda_2(\mathbf{X})}(R\lambda_1(\mathbf{X})\mathbf{x}_1 + T)$

Correspondence $\quad I_1(\mathbf{x}_1) = I_2(h(\mathbf{x}_1))$

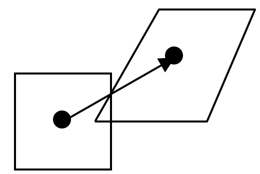
Local Deformation Models

- Translational model



$$h(\mathbf{x}) = \mathbf{x} + d \qquad I_1(\mathbf{x}_1) = I_2(h(\mathbf{x}_1))$$

- Affine model



$$h(\mathbf{x}) = A\mathbf{x} + d \qquad I_1(\mathbf{x}_1) = I_2(h(\mathbf{x}_1))$$

- Transformation of the intensity values taking into account occlusions and noise

$$I_1(\mathbf{x}_1) = f_o(\mathbf{X}, g) I_2(h(\mathbf{x}_1) + n(h(\mathbf{x}_1))$$

Feature Tracking and Optical Flow

- Translational model

$$I_1(\mathbf{x}_1) = I_2(\mathbf{x}_1 + \Delta\mathbf{x})$$

- Small baseline

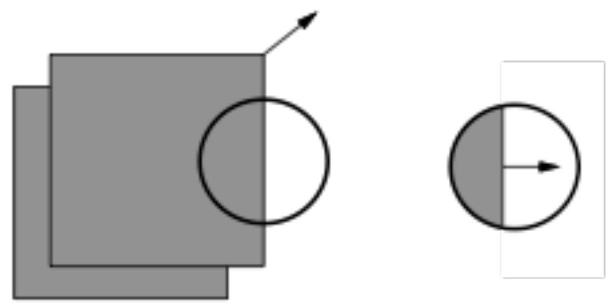
$$I(\mathbf{x}(t), t) = I(\mathbf{x}(t) + \mathbf{u}dt, t + dt)$$

- RHS approximation by the first two terms of Taylor series

$$\boxed{\nabla I(\mathbf{x}(t), t)^T \mathbf{u} + I_t(\mathbf{x}(t), t) = 0}$$

- Brightness constancy constraint

Aperture Problem



- Normal flow

$$\mathbf{u}_n \doteq \frac{\nabla I^T \mathbf{u}}{||\nabla I||} \cdot \frac{\nabla I}{||\nabla I||} = -\frac{I_t}{||\nabla I||} \cdot \frac{\nabla I}{||\nabla I||}$$

Given brightness constancy constraint at single point –
all we can recover is normal flow

Optical Flow

- Integrate around over image patch

$$E_b(\mathbf{u}) = \sum_{W(x,y)} [\nabla I^T(x,y,t)\mathbf{u}(x,y) + I_t(x,y,t)]^2$$

- Solve

$$\begin{aligned} \nabla E_b(\mathbf{u}) &= 2 \sum_{W(x,y)} \nabla I(\nabla I^T \mathbf{u} + I_t) \\ &= 2 \sum_{W(x,y)} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \right) \end{aligned}$$

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \mathbf{u} + \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} = 0$$

$$G\mathbf{u} + \mathbf{b} = 0$$

Optical Flow, Feature Tracking

$$\boxed{\mathbf{u} = -G^{-1}\mathbf{b}}$$

$$G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

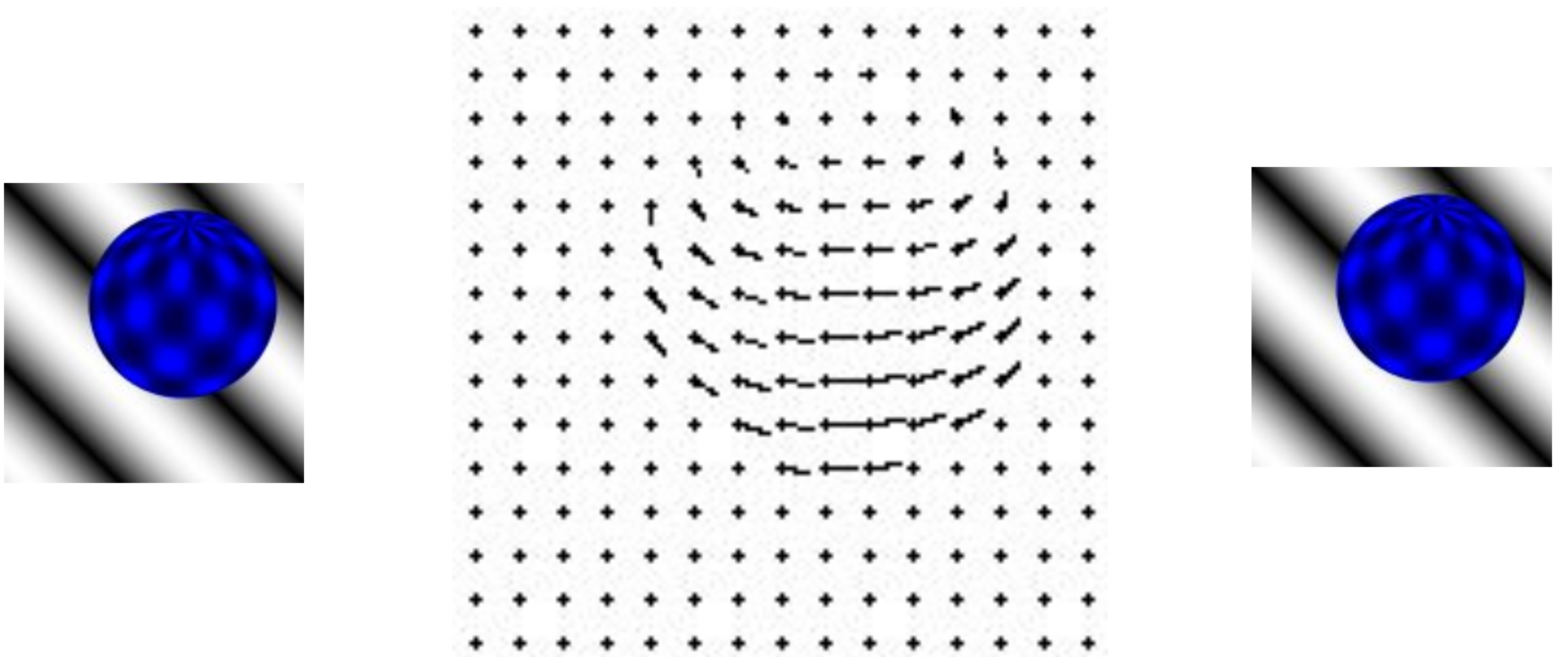
Conceptually:

rank(G) = 0 blank wall problem
rank(G) = 1 aperture problem
rank(G) = 2 enough texture – good feature candidates

In reality: choice of threshold is involved

Optical Flow

- Previous method - assumption locally constant flow



- Alternative regularization techniques (locally smooth flow fields, integration along contours)
- Qualitative properties of the motion fields

Point Feature Extraction

$$G = \left[\begin{array}{cc} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{array} \right]$$

- Compute eigenvalues of G
- If smalest eigenvalue σ of G is bigger than τ - mark pixel as candidate feature point

- Alternatively feature quality function (Harris Corner Detector)

$$C(G) = \mathrm{det}(G) + k \cdot \mathrm{trace}^2(G)$$

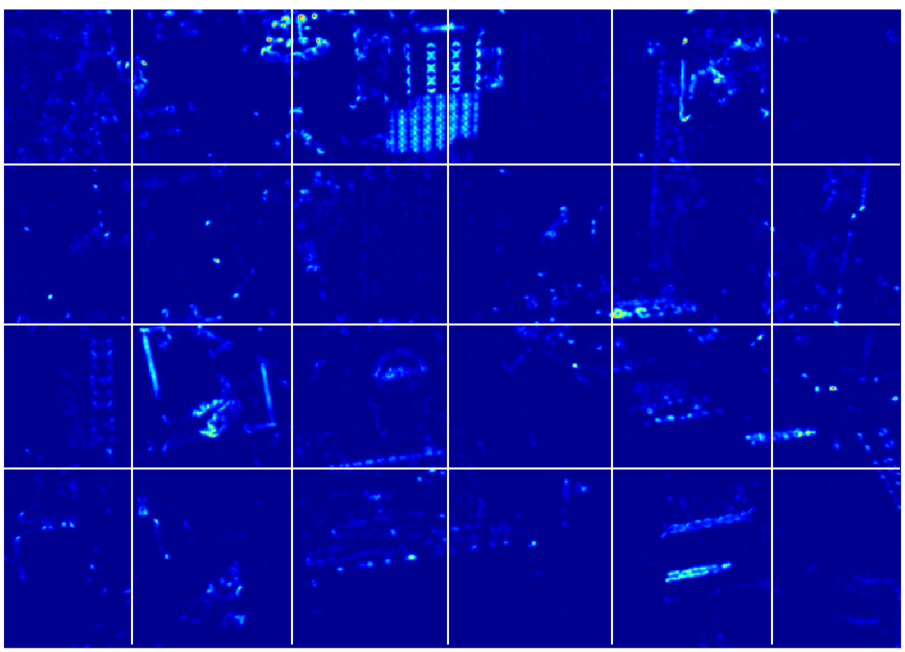
Harris Corner Detector - Example

Feature Selection

- Compute Image Gradient $\nabla I^T = [I_x, I_y]$
- Compute Feature Quality measure for each pixel

$$C(\mathbf{x}) = det(G) + k.\text{trace}^2(G) \qquad G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

- Search for local maxima



Feature Quality Function

Local maxima of feature quality function

Feature Tracking

- Translational motion model

$$E(\mathbf{d}) = \min_{\mathbf{d}} \sum_{W(\mathbf{x})} [I_2(\tilde{\mathbf{x}} + \mathbf{d}) - I_1(\tilde{\mathbf{x}})]^2$$

- Closed form solution

$$\mathbf{d} = -G^{-1}\mathbf{b}$$

$$G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$$

$$\mathbf{b} \doteq \begin{bmatrix} \sum_{W(\mathbf{x})} I_x I_t \\ \sum_{W(\mathbf{x})} I_y I_t \end{bmatrix}$$

- Build an image pyramid
- Start from coarsest level
- Estimate the displacement at the coarsest level
- Iterate until finest level

Coarse to fine feature tracking

1

0

2

1. compute $\mathbf{d}_k = -G\mathbf{b}$
2. warp the window $W(\mathbf{x})$ in the second image by
3. update the displacement $\mathbf{d} \leftarrow \mathbf{d} + 2\mathbf{d}_k$
4. go to finer level $k \leftarrow k - 1$
5. At the finest level repeat for several iterations

Tracked Features